92-05134/

Centre de recherche informatique de Montréal

CRIM / Université de Montréal

Les logiciels d'acquisition des connaissances

-Rapport de projet

CRIM-90/12-01

Claire St-Jacques, conseillère
Centre de recherche informatique de Montréal

Dr Bernard Lefebvre, professeur
Université de Montréal

Catalin Ivan, étudiant
Université de Montréal

Denis Cousineau, étudiant
Université de Montréal

Collection scientifique et technique
décembre 1990

Pour tout renseignement, n'hésitez pas à contacter :

Centre de recherche informatique de Montréal
Centre de documentation
3744, rue Jean-Brillant, bureau 500
Montréal (Québec)
H3T 1P1
Téléphone : 340-5716
Télécopieur : 340-5777

ISBN 2-921316-01-3
Dépôt légal - 4e trimestre 1990
Bibliothèque nationale du Québec
Bibliothèque nationale du Canada
© Centre de recherche informatique de Montréal, 1990

TABLE DES MATIÈRES

Sommaire **5**

Introduction **7**

1. Les approches **9**
1.1 L'approche conventionnelle d'acquisition des connaissances 9
1.2 Traitement attendu d'un logiciel d'acquisition des connaissances 10
1.2.1. Extraction 11
1.2.2. Raffinement 11
1.2.3. Optimisation 13
1.3 Architectures d'un LAC 13
1.3.1. Architecture séparée 14
1.3.2. Architecture intégrée 14
1.4 Acquisition des connaissances selon la nature de l'expertise 15
1.4.1. Cas d'une théorie forte : les arbres de décision 16
1.4.2. Cas d'une théorie faible pouvant se traduire en théorie forte: apprentissage à l'aide d'exemples 21
1.4.3 Cas d'une théorie faible 24
1.5 Format de la base de connaissances générée 24
1.5.1. Règles de production 24
1.5.2. Cas 25

2. Évaluation de sept logiciels d'acquisition des connaissances **27**
2.1 Description des sept logiciels évalués 27
2.1.1 Sophos 28
2.1.2 KAT 31
2.1.3 CAMEO 35
2.1.4 IXL 40
2.1.5 NEXTRA 42
2.1.6 Cogensys Judgment Software 46
2.1.7 CL-Protos 48
2.2 Synthèse comparative des logiciels évalués 51
2.2.1 Critères utilisés pour l'évaluation 51
2.2.2 Tableau comparatif des sept logiciels 54
2.2.3 Conclusion de notre évaluation 56

Conclusion **59**

Glossaire **61**

Références **63**

SOMMAIRE

Le présent rapport décrit les résultats d'une première étude réalisée au Centre de recherche informatique de Montréal (CRIM) sur les outils informatiques ou logiciels d'acquisition des connaissances (LAC) utilisés pour le développement de systèmes experts (SE).

Les activités des laboratoires du CRIM se sont principalement concentrées, jusqu'à maintenant, sur les outils de développement de SE. Toutefois, l'acquisition des connaissances demeure un problème de taille pour lequel un nombre grandissant de logiciels informatiques sont développés. De plus, l'activité cognitive est encore mal structurée et elle nécessite souvent un investissement substantiel de temps et de ressources humaines lors du développement d'un SE.

Suite à ces constats et à une idée originale du Dr Bernard Lefebvre (Univ. de Montréal), de M. Serge Masse (Groupe DMR) et de Mme Claire St-Jacques (CRIM), le CRIM a initié un projet dans le domaine de l'acquisition automatique des connaissances. L'objectif du projet consistait à étudier et à évaluer sur une base comparative un certain nombre de LAC. Ce projet visait surtout à faire un premier "déblaiement" du sujet.

Le choix des logiciels à évaluer s'est effectué d'une façon informelle et non rigoureuse. Nous nous sommes inspirés de publications sur le sujet [15] [16] et d'informations à notre disposition, puis nous avons tenté d'obtenir des fournisseurs des exemplaires gratuits des logiciels. Ceux-ci nous ont été prêtés pour évaluation pendant une courte période (trois mois). Nous avons favorisé des logiciels au coût accessible, utilisables sur des ordinateurs courants et disponibles sur une base commerciale.

Deux étudiants de l'université de Montréal, Messieurs Denis Cousineau et Catalin Ivan, sous la supervision du Dr Bernard Lefebvre, ont réalisé ce projet avec les ordinateurs du CRIM dans le cadre d'un cours terminal du baccalauréat en informatique. Mme Claire St-Jacques a assuré l'organisation (contacts avec les fournisseurs), la gestion et la coordination des activités. Les étudiants ont rédigé un premier rapport qui fut par la suite révisé par le Dr Bernard Lefebvre, Mme Claire St-Jacques et M. Serge Masse. Le 4 mai 1990, dans le cadre de "L'Heure du CRIM", le Dr Lefebvre et ses deux étudiants ont donné une présentation des résultats de ce projet. Enfin, la présente version du rapport résulte d'une révision finale réalisée au CRIM par M. Mario Latendresse et Mme Claire St-Jacques.

Nous tenons à remercier pour leur précieuse collaboration les personnes suivantes : M. Serge Masse (Groupe DMR Inc.) pour ses idées originales et ses commentaires pertinents, M. Mario Latendresse pour la révision du contenu technique du second chapitre, Mesdames Angèle Guérard et Érica Huber qui ont patiemment corrigé et mis en page le présent document, et les membres de l'équipe du support technique du CRIM. De plus, la responsable du centre de documentation du CRIM, Mme Sylvie Tellier, nous a été d'une grande utilité notamment pour effectuer les recherches et références bibliographiques. Enfin, Mme Susan Dubé et M. Jacques Ouellet ont révisé la présente version finale.

Nous ne pouvons prétendre que ce rapport soit une étude prospective approfondie du sujet. Le projet s'est déroulé dans un intervalle de temps relativement court, avec des ressources humaines limitées et à même le budget des activités internes du CRIM. Toutefois, nous espérons que le présent rapport saura fournir aux personnes intéressées par le sujet quelques pistes de réflexion intéressantes !

INTRODUCTION

Les techniques d'exploitation des ordinateurs ont considérablement évolué depuis le codage assembleur jusqu'à l'élaboration des SE (systèmes experts). Mais à chaque technologie correspond un ensemble de problèmes ainsi que des logiciels spécifiques. Les SE demandent la mise au point de techniques d'entrevue, d'acquisition et de représentation des connaissances, d'inférences à partir de règles, etc. Nous nous proposons de présenter une catégorie de logiciels, les LAC (logiciels d'acquisition des connaissances), destinés à faciliter, sinon à automatiser, la phase la plus ardue du développement d'un SE : l'acquisition des connaissances du ou des experts du domaine choisi, suivie de leur transposition sous la forme d'une base de connaissances (BC) utilisable par un outil de développement de SE (GSE).

Le moteur d'inférence et la base de connaissances sont les deux entités de base d'un SE. Si le premier est un logiciel relativement simple à mettre en œuvre, la conception de la seconde n'a, jusqu'à présent, bénéficié d'aucun apport logiciel. La BC est souvent élaborée à la main et sans aide par les cogniticiens et les experts eux-mêmes. L'arrivée des LAC marque donc un tournant considérable.

Le présent rapport tente d'illustrer les possibilités actuelles et futures dans le domaine nouveau des LAC.

Le premier chapitre propose une description des divers besoins, des approches et options possibles pour de tels logiciels.

Le second décrit et évalue spécifiquement quelques logiciels existants. Il se termine par un tableau récapitulatif.

Enfin, dans la conclusion, des possibilités de recherche en ce domaine sont évoquées.

* Note : La définition des abréviations se retrouve dans le glossaire à la fin du rapport.

CHAPITRE I : LES APPROCHES

1.1 L'approche conventionnelle d'acquisition des connaissances

L'approche conventionelle du processus d'acquisition des connaissances ne fait pas appel aux logiciels dont il est question dans ce rapport. En effet, il fait généralement intervenir deux personnes : le cogniticien et l'expert du domaine. Le premier possède la connaissance des logiciels et méthodes utilisées pour générer des SE, le second fournit de façon plus ou moins explicite la connaissance à représenter.

Le transfert des connaissances de l'expert vers le système s'accomplit en plusieurs étapes qui peuvent être itératives. À la phase d'acquisition initiale succède celle de la formalisation des connaissances. Cette dernière permet le passage vers la réalisation d'un prototype implanté à l'aide d'un générateur de SE. Le prototype est ensuite testé à partir de cas, ce qui entraîne généralement une mise à jour des connaissances déjà acquises. Ce cycle est alors itéré en amorçant une nouvelle formalisation jusqu'à ce qu'une certaine stabilité soit obtenue. Il s'agit d'un processus qui peut être très long et pendant lequel le rôle du cogniticien consiste à guider l'élucidation des connaissances de l'expert, à conseiller le choix d'outils appropriés et à aider à la formalisation et à l'implantation de ces connaissances dans le cadre des logiciels choisis.

Cette intervention peut susciter certains problèmes parmi lesquels nous pouvons généralement identifier les suivants [1] [10] [20] :

- Problème de compréhension : l'expert et le cogniticien ont à l'origine des compétences différentes, souvent complémentaires et disjointes. Afin d'obtenir une compréhension mutuelle, indispensable à la réalisation du système, chacun d'eux doit faire l'effort d'acquérir un minimum des compétences de l'autre, ce que les intervenants ne sont pas toujours en mesure de faire.

- Problème de disponibilité : lors du processus d'acquisition, de nombreuses rencontres doivent être planifiées entre l'expert et le cogniticien, mais l'un et l'autre n'ont pas toujours la disponibilité nécessaire.

- Problème de paternité du produit créé : l'expert éprouve souvent le sentiment d'être dépossédé de son savoir au profit du cogniticien. Il peut alors réagir négativement en

dissimulant certains éléments importants ou en introduisant certaines inconsistances. À l'inverse, il peut aussi craindre de révéler la simplicité de ce savoir et donc être tenté de compliquer artificiellement les solutions qu'il propose.

- Problème de maintenance : un SE est un produit évolutif qui doit suivre les exigences de l'expert et des utilisateurs. Les modifications nécessitent une nouvelle intervention conjointe du cogniticien et de l'expert. Cet exercice peut déplaire à l'expert, d'autant plus qu'il ne se perçoit pas toujours comme l'auteur ou l'inventeur du produit. Le cogniticien, quant à lui, peut ne pas apprécier d'avoir à raffiner un système qu'il croyait au point et dont il peut avoir oublié le contexte.

L'utilisation d'un LAC bien conçu devrait limiter l'intervention du cogniticien et, par conséquent, réduire les problèmes mentionnés précédemment. La section suivante expose les trois étapes successives qu'un LAC devrait couvrir.

1.2 Traitement attendu d'un logiciel d'acquisition des connaissances

Un logiciel ayant pour but d'acquérir le savoir d'un expert humain devrait procéder en trois phases successives : l'extraction, le raffinement et l'optimisation [1] [10] [17] [18] [20].

Un LAC est principalement utilisé par l'expert, que ce soit directement ou par l'intermédiaire d'un cogniticien.

Dans les deux premières phases, le LAC et le cogniticien accomplissent une sorte de maïeutique. Cette dernière est la méthode employée par Socrate pour inciter l'interlocuteur (ici l'expert) à dévoiler des concepts qu'il possède mais dont il n'a pas nécessairement conscience, le menant ainsi à une meilleure compréhension de son savoir.

Pour donner un exemple concret, imaginons que nous voulons connaître le corps humain. La première phase consiste à séparer le corps selon les distinctions visibles à priori : mains, jambes, corps, tête, articulations, etc. La seconde phase vise à découvrir l'organisation interne : nous avons alors une vision plus spécifique des ensembles de muscles, des nerfs et du fonctionnement des organes. Enfin, l'optimisation du savoir est de comprendre le codage génétique qui décrit naturellement tout le corps humain : nous obtenons ainsi des

connaissances différentes des représentations à priori du corps, mais elles sont cependant aussi complètes et beaucoup plus condensées.

Nous présentons ci-dessous une description des trois phases de l'acquisition des connaissances : l'extraction, le raffinement et l'optimisation.

1.2.1. Extraction

L'extraction consiste à construire un squelette des connaissances nécessaires dans le domaine. Cette phase permet d'acquérir la terminologie et la structure fondamentale de la base de connaissances. Elle doit aussi permettre d'identifier les buts ou les solutions recherchées, de même que les données permettant de les obtenir. Enfin, cette phase définit les caractéristiques permettant de décrire les faits et les problèmes du domaine.

Divers chercheurs maintiennent que cette étape doit s'effectuer de façon systématique et structurée afin de couvrir entièrement le domaine : elle doit, d'une part, écarter les oublis potentiels de facteurs importants et, d'autre part, éviter d'approfondir à l'excès les connaissances dans leurs détails.

Pour accomplir cette phase avec succès, il est nécessaire de suivre une méthodologie d'interrogation de l'expert. Cette méthodologie peut être générale, applicable à toutes les connaissances dans des domaines très différents les uns des autres : la médecine, l'exploration minière, la conception d'algorithmes aussi bien que l'entretien d'un réacteur nucléaire. La méthodologie peut, au contraire, être spécifique à un domaine particulier, ce qui rend le LAC plus facile à développer grâce à l'existence d'un vocabulaire standardisé, de publications ou manuels déjà existants, etc.

1.2.2. Raffinement

La phase de raffinement a pour but d'étendre progressivement l'ensemble des connaissances de la base. D'une façon imagée, le LAC doit habiller de "chair" le squelette mis à jour lors de l'extraction. Ceci consiste à recueillir le maximum de détails possible tout en organisant ce savoir de manière à éviter les contradictions, à relier tous les sous-domaines et sous-buts entre eux, à accomplir une certaine validation et à faire ressortir l'explication des étapes du raisonnement. Cette phase peut se faire de deux manières

distinctes : en interrogeant l'expert toujours plus spécifiquement ou en analysant un ensemble de cas fournis par celui-ci (problèmes, solutions apportées par l'expert, explications ou justifications éventuelles).

Puisque l'expertise n'est pas toujours formalisable, il est difficile d'utiliser une procédure descendante. Il existe actuellement deux approches développées par divers chercheurs : l'analyse statique et l'analyse dynamique.

- L'analyse statique de la base de connaissances permet de déceler des modes de raisonnement (d'inférence) trop faibles (par exemple en termes de facteurs de confiance) ou ne semblant pas mener pas à des conclusions utiles. Il est possible ainsi de questionner l'expert et de trouver les informations (manquantes ou erronées) qui font que le LAC ne réussit pas à reproduire le raisonnement correctement.

- L'analyse dynamique utilise la base de connaissances pour résoudre divers cas-tests et, en comparant ses solutions avec celles de l'expert, elle trouve les erreurs dans le raisonnement du LAC ainsi que les omissions qui les ont provoquées. Ces cas-tests peuvent provenir de l'expert ou être générés par le logiciel selon les connaissances acquises.

Dans les deux cas, le logiciel doit être en mesure d'expliquer clairement les échecs de son raisonnement, du moins suffisamment pour que l'expert interrogé puisse ajouter les informations qui empêchaient le système d'effectuer les bonnes déductions. Le processus d'acquisition, par interrogation ou par analyse de cas réels, se prolonge jusqu'à ce que l'expert soit satisfait.

Il peut aussi arriver que plusieurs experts soient disponibles pour élucider un domaine donné, ou qu'une méthodologie de travail en groupe soit utilisé. Le bénéfice gagné avec plusieurs points de vue différents peut être un atout important, à condition que le logiciel d'acquisition soit en mesure de mettre de l'ordre parmi les approches ou les vocabulaires pouvant diverger d'un expert à l'autre. Il est donc impératif de posséder un mécanisme de coordination qui, par exemple, ne "s'emballe" pas devant plusieurs règles contradictoires, mais qui reconnaît le contexte spécifique que demande l'application de chacune d'elles.

Dans cette phase, il importe de considérer les conséquences de l'ajout de nouvelles connaissances. En effet, une nouvelle information peut affecter des inductions correctes en même temps qu'elle peut en corriger d'autres.

Enfin, puisque c'est ici la phase où l'expert dévoile ses secrets et son intuition, il est important d'avoir une technique d'acquisition très souple qui traite humainement l'expert. Il est vrai que la machine peut être utile à l'expert en lui faisant prendre conscience des aspects cachés de son expertise. Toutefois, cette phase fait appel à un compromis entre essayer de suivre à la lettre le raisonnement de l'expert et imposer à celui-ci une structure rigoureuse qui risque de l'ennuyer (par exemple, l'énumération d'un grand nombre de possibilités risque d'irriter l'expert si, pour lui, il est évident que seule une minorité est digne d'intérêt).

En demandant à l'expert de formaliser de plus en plus ses connaissances, des informations inconnues sur le domaine peuvent être révélées, mais par contre, l'expertise en augmentant dépend davantage de l'intégration et de la fusion des connaissances disparates. H. Dreyfus [8] affirme que, si le novice agit formellement en fonction de règles apprises, l'expert, lui, agit de manière inconsciente et globale, quelquefois en anticipant ou en élaborant un plan global.

1.2.3. Optimisation

La dernière phase de l'acquisition des connaissances, l'optimisation, a pour but de transformer la BC développée en une application plus efficace et plus performante pour la résolution de problèmes. L'expert n'y participe pas, cette tâche étant celle du cogniticien, d'un logiciel approprié ou de la combinaison des deux.

La base de connaissances est analysée, reformulée et compilée; éventuellement des procédures algorithmiques peuvent y être intégrées afin de supprimer des éléments superflus ou de simplifier des raisonnements tortueux.

1.3 Architectures d'un LAC

Nous avons observé deux types d'architecture : l'architecture séparée et l'architecture intégrée.

1.3.1 Architecture séparée

Dans une architecture séparée, les deux activités d'acquisition des connaissances et d'exploitation par un SE de la base de connaissances sont explicitement distinctes. D'une part, le LAC crée la base de connaissances (BC) et d'autre part, la BC est mise en œuvre par un second logiciel, le générateur de systèmes experts (GSE) :

LAC --> BC --> GSE.

Cette architecture a l'avantage d'être générique puisqu'un LAC peut servir à produire des BC pour divers GSE.

Toutefois, des problèmes peuvent apparaître lors de la construction et de la maintenance de la BC. Il peut en effet être nécessaire d'effectuer de fréquents passages du LAC au GSE pour tester et valider la BC en cours de conception. La maintenance de la BC ne peut être dynamique, puisqu'elle doit être raffinée et "recompilée" à chaque modification. Pour remédier à cela, nous pouvons évidemment imaginer un éventuel ensemble de logiciels intégrés, mais il est nécessaire pour ce faire de standardiser les formats.

Notons par ailleurs que le LAC peut être équipé d'outils de visualisation des connaissances (cartes, graphes, arbres, etc.) qui facilitent la mise au point des connaissances et donnent une meilleure compréhension des inférences ou raisonnements. Bien que ces outils apportent une facilité accrue de développement et de maintenance de la BC, ils n'éliminent pas pour autant la nécessité de tester la BC au sein du GSE dans des situations réelles.

1.3.2 Architecture intégrée

Les logiciels à architecture intégrée incorporent le LAC et le GSE au niveau de la BC :

LAC = GSE <---> BC.

Dans ce cas, la distinction entre l'élaboration de la BC et son utilisation pour la résolution de problèmes n'existe plus. Le passage transparent de l'une à l'autre ne cesse que lorsque les résultats aux tests sont positifs et que l'utilisateur est satisfait.

Un tel système est plus flexible puisque la BC est toujours prête à être utilisée de même qu'à être mise à jour. L'entretien est plus dynamique puisque les modules de validation et

de vérification des connaissances, par exemple ceux assurant le maintien de la consistance entre les règles ou de la hiérarchie entre les sous-buts, peuvent fonctionner de pair avec le GSE. De plus, il est facile d'étendre et de raffiner graduellement le SE en ajoutant simplement des informations en cours d'utilisation. Enfin, il est clair que cette architecture diminue la possibilité d'ennuis dus à la multiplicité des logiciels. Ce type de système exige un nombre limité de conversions, de passages entre les logiciels formant le LAC et le GSE, de manipulations au niveau des fichiers, etc.

Un élément distinctif par rapport à l'architecture séparée concerne la responsabilité de l'utilisateur. Si le système est laissé en évolution, il continue à apprendre au fur et à mesure des sessions d'utilisations; il faut donc s'assurer que les utilisateurs soient également des experts. À moins que les modules de maintenance de la consistance ne soient particulièrement sophistiqués, il ne faut pas oublier que le logiciel peut aussi adopter des mauvaises habitudes ou apprendre des raisonnements faux.

1.4 Acquisition des connaissances selon la nature de l'expertise

Nous présentons dans cette section une classification de la nature de l'expertise, classification sur laquelle peut être basé le choix d'un LAC adéquat pour l'application et le domaine qui nous intéressent.

Nous établissons une distinction entre l'expertise à théorie forte et l'expertise à théorie faible. Une expertise à théorie forte permet d'utiliser des techniques symboliques ou mathématiques pour représenter les divers aspects du domaine. Par exemple, l'expert peut rendre compte de plusieurs situations en décrivant des règles et des heuristiques traitant celles-ci; il est, dans ce cas, en mesure de décrire le domaine explicitement et clairement, comme par exemple à l'aide d'arbres de décision. Par contre, une expertise à théorie faible ne possède pas de formalisme décrivant plusieurs situations; elle emploie plutôt l'intuition, l'expérience ou l'analogie pour traiter chacune d'elles. Dans la plupart des cas, ce dernier type d'expertise dépend de l'image que chaque expert se fait du domaine.

Il existe des types d'expertise qui ont recours simultanément à la théorie forte et à la théorie faible. Dans un tel contexte, l'expert utilise des règles pour certaines situations et favorise l'expérience et l'intuition pour d'autres; ou encore, il peut pour certaines phases de raisonnement utiliser l'expérience (implicite) et, pour d'autres, employer des méthodes rigoureuses et explicites. Toutefois, nous nous contentons dans ce rapport d'effectuer une

première classification des LAC et nous utilisons uniquement les termes "théorie forte" et "théorie faible".

Dans le cas d'une expertise à théorie forte, l'expert est en mesure d'exprimer et de formaliser les notions propres au domaine d'application. Le LAC peut donc se contenter d'offrir des outils qui guident l'expert et lui permettent de structurer la construction de la base de connaissances.

Si l'expertise est à théorie faible, le LAC se doit d'être encore plus évolué et de procurer un mécanisme d'apprentissage qui va permettre de formaliser les connaissances. Nous avons alors le choix entre deux possibilités : obtenir une formalisation compréhensible par l'humain ou une formalisation utilisable uniquement par le LAC ou un GSE. Dans ce dernier cas, la base de connaissances n'est pas directement compréhensible pour l'humain; par exemple, c'est le cas des techniques de codage de la base de connaissances par l'indexation de cas.

Nous obtenons ainsi trois types d'expertise auxquels correspondent diverses catégories de LAC : l'expertise peut être à théorie forte, à théorie faible, et à théorie faible pouvant se traduire en une théorie forte compréhensible. Nous examinons ces trois catégories dans les sections suivantes.

1.4.1 Cas d'une théorie forte : les arbres de décision

Il est ici possible de créer un arbre décrivant l'enchaînement des attributs vers le but ou un réseau dans lequel sont reliés les attributs, les faits et les questions.

Dans un arbre de décision (voir figure 1), une structuration hiérarchique des connaissances est développée à partir de la racine, où se trouve le point de départ du problème, vers les branches qui représentent chacune un attribut concourant au but ou au sous-but recherché. À chaque descente de niveau, la valeur d'attributs est déterminée; aux feuilles, une solution est identifiée. Nous parlons alors de spécialisation : à partir des faits généraux, nous progressons vers les faits de plus en plus spécifiques. Autrement dit, le nombre de connaissances (d'attributs) augmente afin de différencier de plus en plus les solutions.

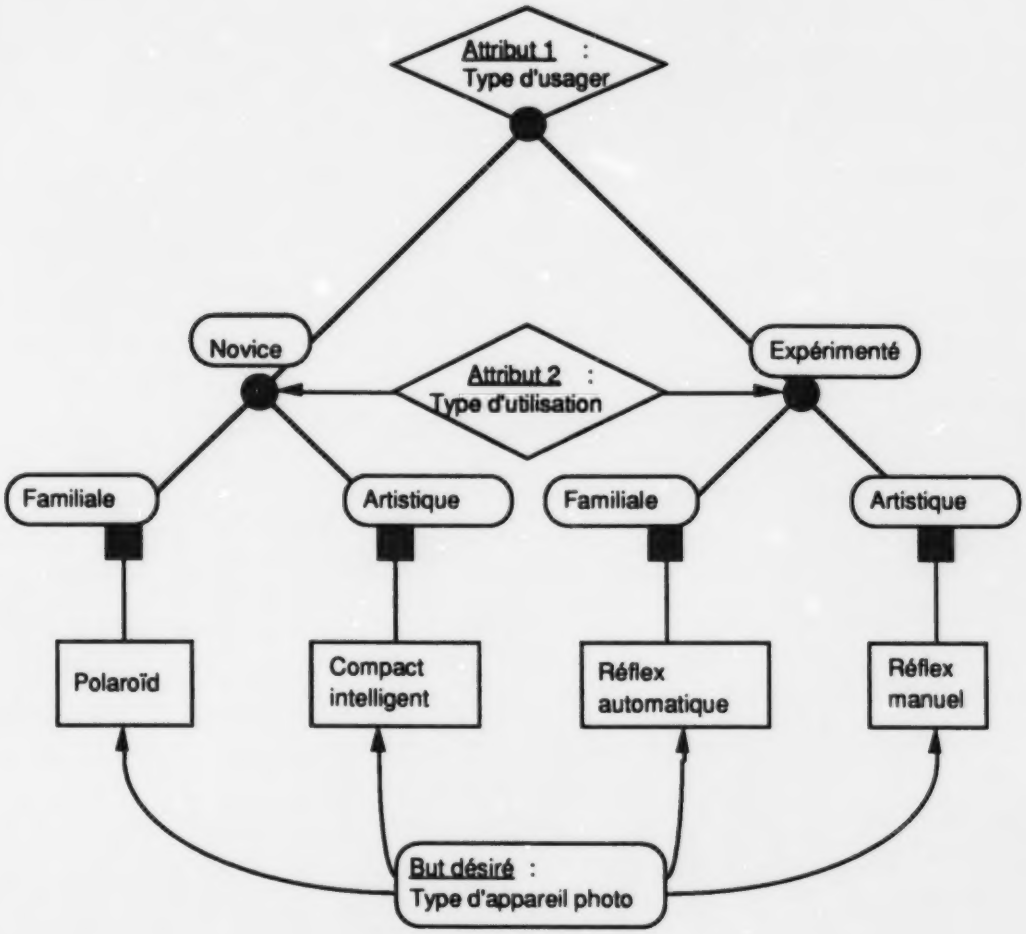


Figure 1 : Arbre hiérarchique illustrant la combinaison d'attributs pour la distinction de quatre appareils-photos.

Une telle représentation en arbre de décision capture difficilement le savoir heuristique. Elle est adéquate si l'expertise est à théorie forte; l'expert doit de plus être capable d'expliquer précisément son raisonnement, de ne rien oublier, de ne pas nuancer, ce qui est rarement le cas [8]. L'arbre ne peut rien "connaître" ou "apprendre" en dehors des options déterminées par ses branches .

Les règles sont générées sans plus de raffinement à partir des feuilles, mais cette stratégie n'est pas efficace ou optimale. Il peut en effet y avoir des redondances inutiles dans l'arbre. Par exemple, un attribut qui est pertinent seulement pour 5% des solutions est

vérifié dans toutes les règles, mais alors 95% des règles vont se répéter avec des différences minimes sur les valeurs de cet attribut (règles appartenant à une même branche de l'arbre). Ces répétitions de règles similaires sont inutiles car elles mènent à une seule et même solution [5].

Idéalement, le nombre de règles doit être aussi restreint que possible. De plus, le système doit identifier le chemin le plus direct vers le but, utiliser le nombre minimum d'attributs pour ce faire et ne doit pas demander plus d'une fois ou inutilement un renseignement (refaire les mêmes calculs ou recherches).

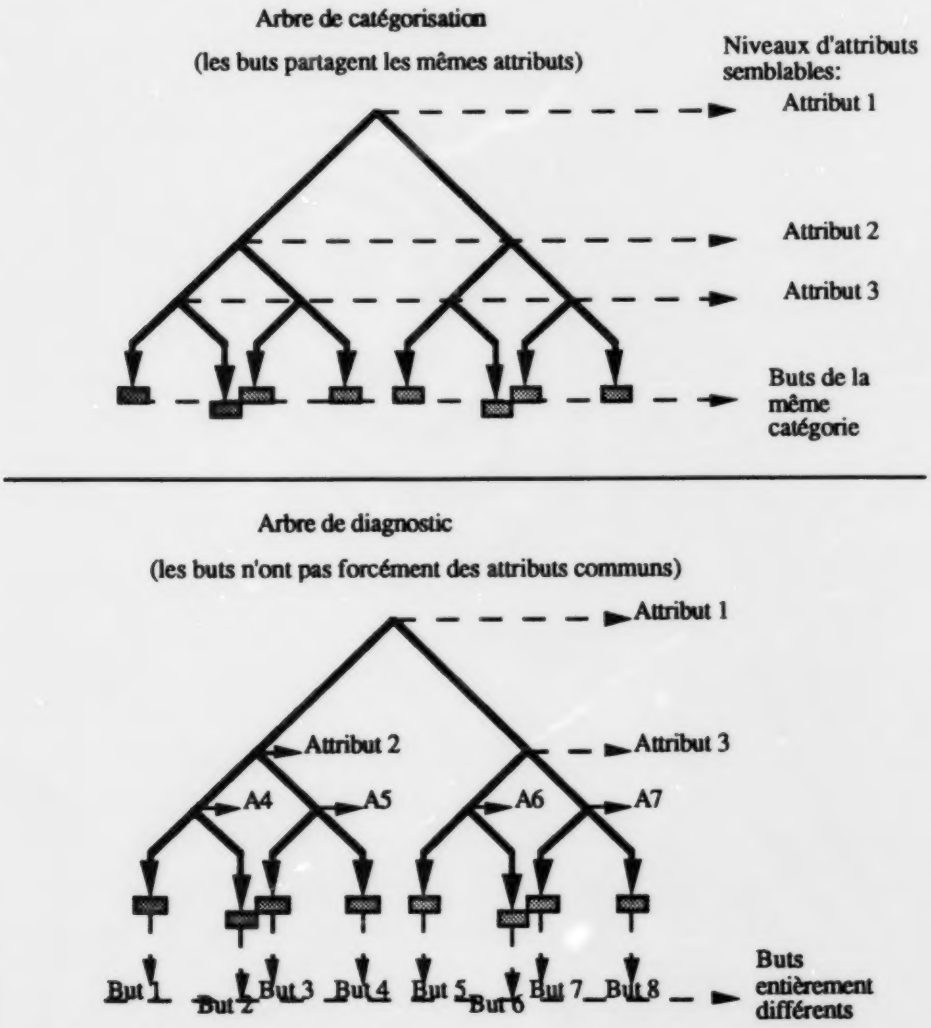


Figure 2 : Différence entre catégorisation et diagnostic

La figure 2 illustre la différence généralement observée entre un arbre de catégorisation et un arbre de diagnostic. Dans le premier cas (catégorisation), le but à atteindre est clair : il s'agit de classer le nouveau cas fourni à l'aide d'une combinaison d'attributs connus et exhaustifs, généralement disponibles et tous pertinents. Dans un arbre de diagnostic, le but n'est pas toujours connu de même que les attributs pertinents; les attributs à considérer varient en fonction du but à atteindre. Un arbre de décision similaire à celui présenté à la figure 1 risque de n'être pas adéquat pour les problèmes de diagnostic.

Mentionnons enfin que, puisqu'un arbre décrit exhaustivement l'ensemble des cas possibles, il peut être fastidieux pour l'humain de les considérer tous.

Il est certes possible d'optimiser un arbre de décision en un réseau compact de règles [23], mais il est aussi possible de tracer directement un réseau en demandant à l'expert d'identifier les faits à connaître et les liens qui les enchaînent (voir figure 3). Un tel réseau est plus flexible : il ne décrit plus les solutions à partir de l'énumération et de la combinaison des attributs, mais traduit plus fidèlement la réalité en donnant à l'expert la liberté de relier les nœuds (attributs, faits, questions, etc.) comme il veut [3].

Aucune des deux représentations, arbre ou réseau, ne garantit la totale transmission de l'expertise car il est toujours possible que l'expert oublie un attribut ou un lien. Le réseau possède un pouvoir d'expression plus flexible, mais délègue davantage de responsabilités à l'expert qui doit pouvoir fournir une représentation exhaustive et correcte de son savoir.

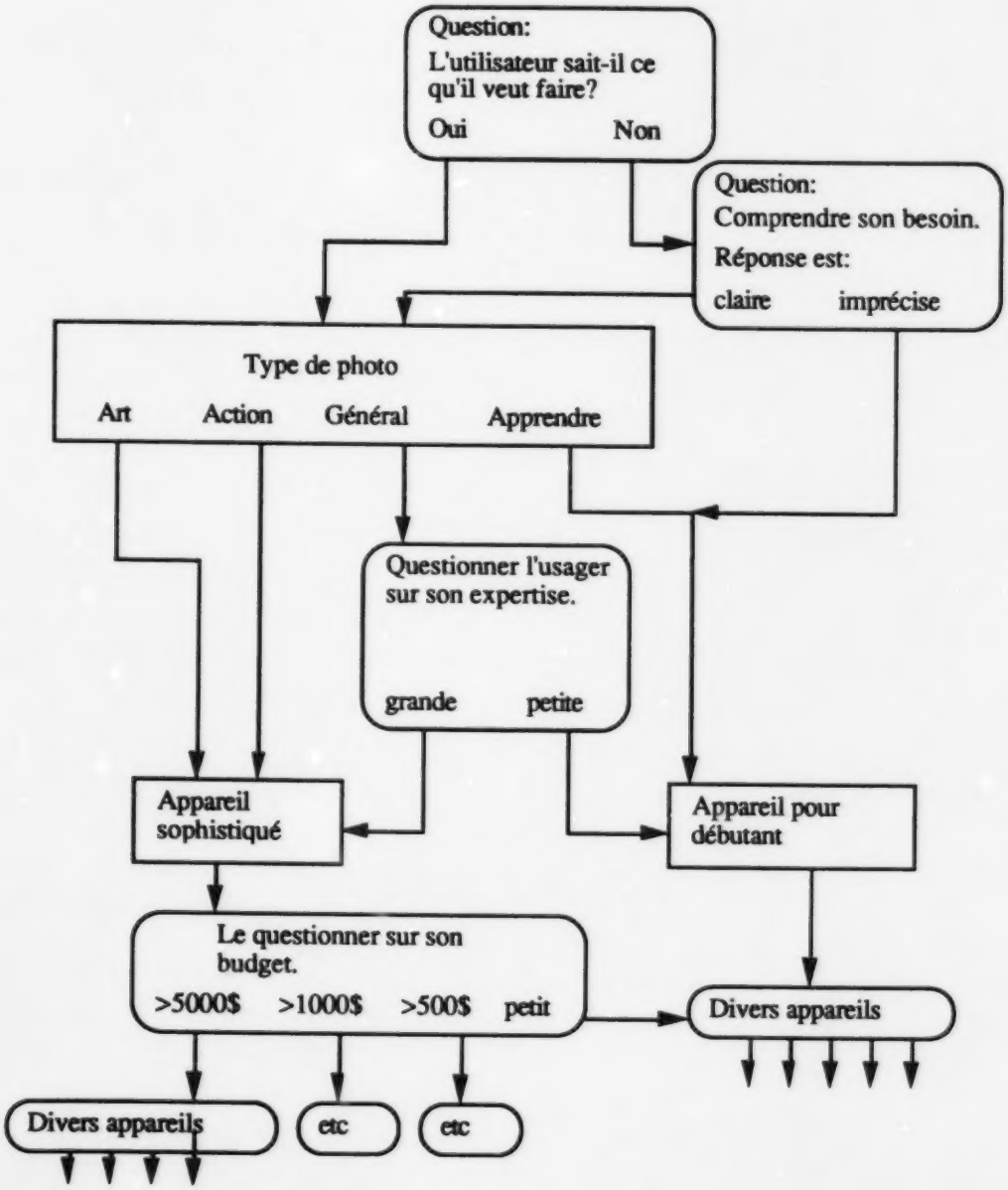


Figure 3 : Exemple de réseau de décision pour le choix d'un appareil-photo

1.4.2 Cas d'une théorie faible pouvant se traduire en théorie forte : apprentissage à l'aide d'exemples

Ce type d'expertise n'est pas préalablement formalisée et le LAC doit permettre de le faire sous une forme soit compréhensible directement par l'expert humain, soit simplement utilisable par le LAC ou le GSE.

Plusieurs catégories de logiciels peuvent être utilisées dans ce cas. La plus élémentaire est formée de ceux qui, à partir de règles formulées en vrac par l'expert, organisent cette

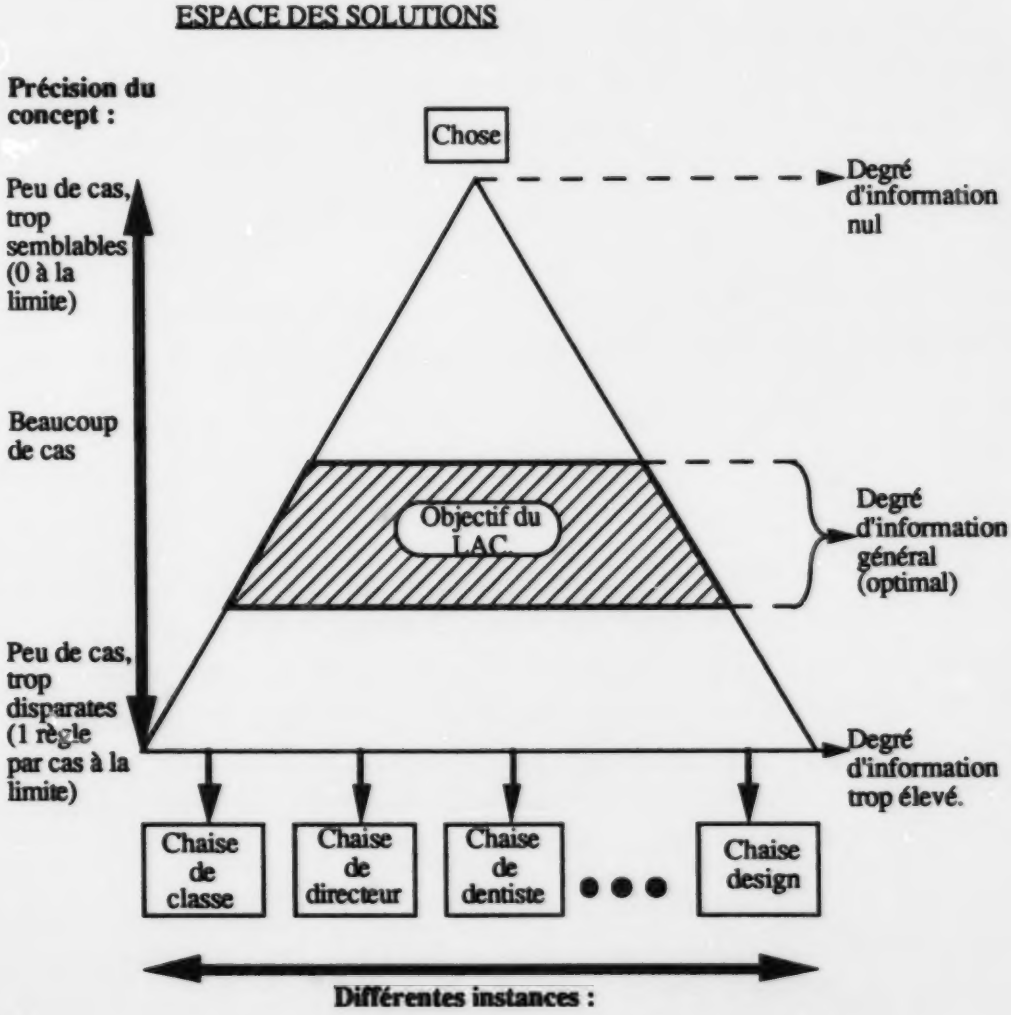


Figure 4 : Généralisation à partir de cas (classification d'une chaise)

connaissance et construisent un arbre de décision en choisissant à chaque niveau l'attribut le plus pertinent. Les autre catégories mettent en œuvre des procédés plus élaborés.

Imaginons qu'au lieu de décrire et d'organiser les attributs et les sous-buts qui mènent à une solution donnée, le cogniticien demande à l'expert de remplir des formulaires qui représentent des cas réels [1] [2] [10]. Par exemple, un expert médical fournit les données suivantes : identification du patient, âge, troubles observés, diagnostic donné par l'expert, actions prises par l'expert. Ce dernier n'a pas alors à expliciter le raisonnement qui le mène aux diverses conclusions. Dans ce type d'acquisition, un grand nombre de cas, sous forme de paires symptôme/solution, est présenté au LAC qui cherche à trouver les concepts par diverses voies d'analyse, statistiques ou autres.

Nous parlons de généralisation lorsque la méthode identifie les traits communs à diverses solutions et les utilise pour différencier les classes de cas; la figure 4 illustre la nature du travail à faire dans l'espace des cas analysés [19]. Nous parlons de spécialisation si, au contraire, nous débutons avec des règles très générales qui discriminent peu, mais qui sont graduellement raffinées pour rendre compte des cas observés. Ce processus implique la nécessité pour le logiciel de s'adapter chaque fois qu'un cas résolu introduit lors de cette phase d'apprentissage provoque une situation d'exception non prévue.

Ce type d'apprentissage à partir de cas peut se faire de manière progressive (chaque cas modifie les connaissances) ou globale (les conclusions sont tirées à partir de l'ensemble des informations).

L'utilisation de grilles d'attributs (voir figure 5) est une autre méthode inspirée d'une technique psychométrique [10] [17] [18]. Elle consiste à remplir une table contenant les attributs sur un axe et les buts sur l'autre. Chaque but (ou solution correspondant à un cas) est décrit en termes de valeurs relatives des attributs (selon une gradation allant de 1 à 5 dans le cas de la figure 5). Une fois que la table est remplie pour chaque solution possible, le LAC est en mesure de découvrir un ensemble efficace de règles décrivant les raisonnements impliqués par les cas présentés.

Cette approche, aussi appelée par induction, est plus créative car elle nécessite beaucoup moins de formalisation et d'explications de la part de l'expert; il suffit au LAC d'observer l'expert au travail pour que le système puisse tirer des conclusions et découvrir des relations. De plus, en observant et en captant "sur le vif" le travail de l'expert, il est moins probadble que celui-ci oublie de formaliser des éléments de connaissances secondaires.

Attributs

Cas	Expérience de l'utilisateur	Fréquence d'utilisation	Photo familiale	Photo d'art ou sportive
Polaroïd	1	4	5	0
Compact	1	3	4	2
Reflex automatique	3	3	3	3
Reflex manuel	5	4	5	4

L'échelle de notation s'étend de 1 à 5

Figure 5 : Exemple de grille d'attributs

1.4.3 Cas d'une théorie faible

Les LAC supportant ce type d'expertise sont les plus généraux. Par contre, ils se comportent souvent comme des boîtes noires puisqu'ils ne sont pas aptes à extraire une théorie sous-jacente et donc à expliquer les moyens et le raisonnement utilisés pour arriver à une conclusion. Ils fonctionnent par analogie et leur analyse consiste à classer et à comparer les cas ou problèmes à résoudre avec ceux déjà connus.

Bien que nous n'en ayons pas évalué de représentants, il existe un autre mode d'apprentissage par cas qui entre dans cette catégorie : l'entraînement d'un réseau neuronal. Un tel réseau peut apprendre à résoudre un problème donné et est surtout utilisé pour des problèmes de catégorisation et de reconnaissance. L'utilisation de réseaux neuronaux (ou modèles connexionistes) pour un LAC est particulière en ce sens que les connaissances acquises ne sont pas explicites, mais plutôt diffusées dans les schémas d'activation des neurones. Ce système profite peu d'un savoir déjà formalisé puisqu'il doit absolument être entraîné cas après cas, ce qui est long. Il s'agit ici d'un logiciel à architecture très intégrée qui exploite lui-même le savoir acquis.

1.5 Format de la base de connaissances générée

Les LAC présentés ont pour but d'amasser un savoir, de manière explicite ou implicite. Dans leur forme intermédiaire ou finale, les connaissances peuvent être générées et organisées comme des règles de production ou des cas. Le format que leur donne le LAC conditionne le type de traitement et les performances.

1.5.1 Règles de production

Cette option, dans laquelle les connaissances sont générées sous forme de règles de production, est celle de la plupart des LAC qui opèrent au bénéfice des GSE traditionnels. La BC est formée de règles, souvent accompagnées de facteurs de confiance. Un moteur d'inférence est utilisé pour construire une chaîne d'inductions qui simule le raisonnement et mène à la solution recherchée [19].

1.5.2 Cas

La BC consiste en un ensemble de cas qui peuvent être reliés ou non. Quand ils le sont (voir figure 6), il s'agit d'un réseau de descriptions de cas interreliés par des index de ressemblances spécifiques ou générales, ou bien par des index des différences qui établissent une distinction entre les solutions (apprentissage par généralisation). Si les cas ne sont pas reliés, la BC devient une liste qui est parcourue lors de sa mise en œuvre; une procédure de reconnaissance des formes, appelée apprentissage par analogie, est utilisée pour trouver les meilleures solutions.

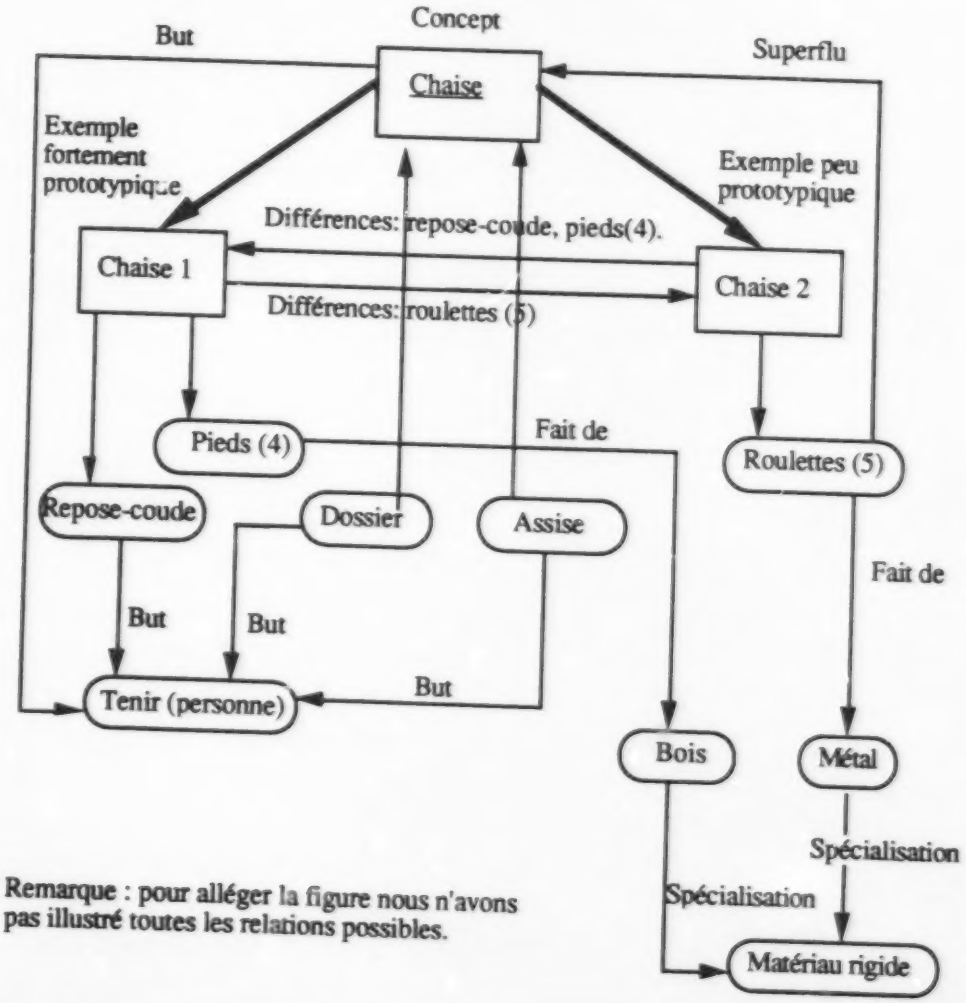


Figure 6 : Exemple d'un réseau de cas pour le concept de chaise.

Nous considérons que l'induction à partir de cas présentés et l'organisation des cas en un réseau selon les liens de ressemblance/différence est une approche plus complète et globale, dans le sens où elle implique un "non-déterminisme" de la BC, une grande souplesse dans l'interprétation de la réalité et une capacité évolutive de la BC.

Nous plaçons dans cette catégorie les réseaux neuronaux (ou modèles connexionistes); nous considérons que les ensembles d'activation des neurones représentent un genre de prototype, même s'il est très difficile de reconnaître explicitement les données et les traitements effectués.

CHAPITRE II : ÉVALUATION DE SEPT LOGICIELS D'ACQUISITION DES CONNAISSANCES

La première section décrit de façon détaillée les sept logiciels que nous avons étudiés. La seconde comporte des tableaux récapitulatifs résumant l'ensemble de notre évaluation, expose les critères utilisés, et présente notre conclusion sur ces logiciels.

2.1 Description des sept logiciels évalués

Dans cette première section, nous donnons un aperçu de chacun des sept logiciels d'acquisition de connaissances mis à la disposition des auteurs. Les sept logiciels sont :

- Sophos
- KAT
- IXL
- CAMEO
- NEXTRA
- Cogensys Judgment Software
- CL-Protos.

Les cinq premiers sont des logiciels qui produisent des règles, bien que NEXTRA procède à partir de cas; les deux derniers travaillent sur des cas, d'où l'ordre de présentation que nous avons adopté.

Pour chacun des logiciels, nous discutons dans un premier temps de données matérielles : le coût, les ordinateurs qui supportent le logiciel, les domaines d'applications, etc. Ensuite, nous présentons le modèle adopté par le logiciel : l'architecture, le type d'expertise, le format de la base de connaissances générée. Comme nous le verrons, ces données sont très importantes afin de savoir si le logiciel est approprié à l'application que nous envisageons. Nous étudions ensuite les entrées-sorties ainsi que les traitements intermédiaires qu'exécutent ces logiciels, car il importe avec des connaissances symboliques d'en avoir une bonne vue locale et globale. Nous terminons avec la convivialité du logiciel, à savoir la facilité d'apprentissage de son fonctionnement, de son utilisation et autres.

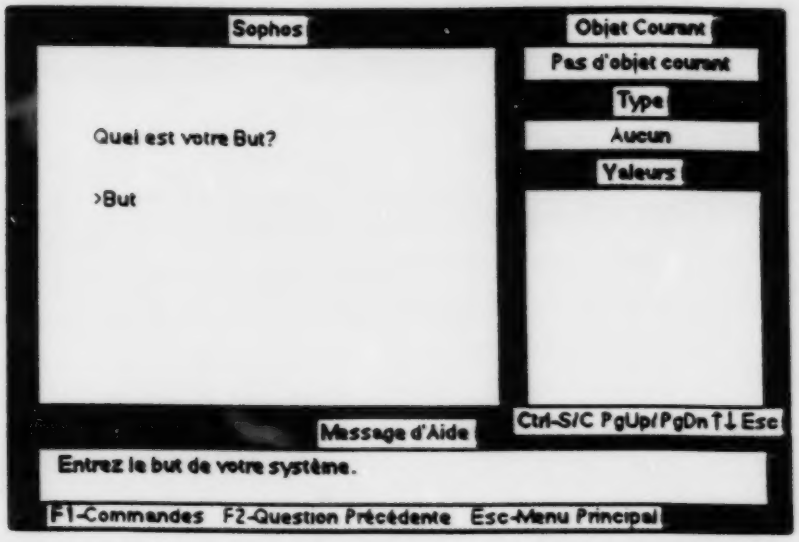
2.1.1 Sophos

Le logiciel Sophos [26] est développé et commercialisé par l'entreprise montréalaise Cognisys, spécialisée dans l'élaboration de bases de connaissances à l'aide d'un cogniticien (*knowledge engineer*). Pour supporter le travail du cogniticien, cette compagnie a choisi de développer un logiciel que l'expert peut utiliser pour constituer une première base de connaissances sur laquelle le cogniticien peut travailler (c.-à-d. qu'il peut raffiner et optimiser).

Ce logiciel, en vente pour 1 500 $, fonctionne sur les micro-ordinateurs IBM équipés d'une carte graphique et ne nécessite rien de plus qu'une version du DOS. Ce logiciel a déjà été utilisé, entre autres, pour élaborer une base de connaissances dans le domaine des prêts bancaires (fournie comme démonstration avec le logiciel). Tout porte à croire qu'il peut être utilisé pour construire une base de connaissances dans des domaines très divers.

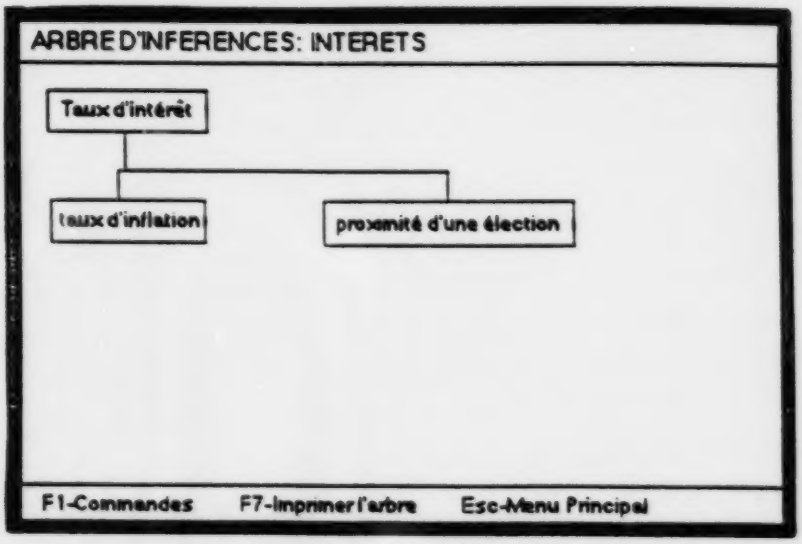
Ce logiciel est basé sur une architecture séparée, c.-à-d. que sa seule fonction est de construire une base de connaissances, mais il n'est pas en mesure de la mettre en œuvre dans un contexte d'aide à la décision (le but des SE). Sophos produit donc des règles qui doivent être exportées vers un GSE. Notons que Sophos peut exporter vers Level5, Intelligence Service 2 et VPExpert, et il est même possible de créer une application qui va traduire sa sortie en format texte pour d'autres GSE.

Sophos, pour produire ses règles, va demander à l'expert les sous-buts à réaliser pour atteindre son but et ce, récursivement pour chaque sous-but jusqu'à ce que l'information soit directement accessible (voir figure 7). Il s'agit donc de construire un arbre d'inférence (voir figure 8). Cette structure de connaissance convient lorsqu'il s'agit de classifier un nouveau cas en parcourant un ensemble de nœuds à partir de la racine de l'arborescence jusqu'à une feuille. Elle est toutefois moins efficace pour les problèmes de diagnostic où, comme nous l'avons mentionné à la section 1.4.1, le but à atteindre varie en fonction des éléments d'information. En effet, l'arbre devient alors très profond et le parcours de chaque nœud (attribut) de l'arbre d'inférence n'est pas nécessairement pertinent.



© Copyright Cognisys Consultants Inc., 1989, 1990.

Figure 7 : Entrée de buts et sous-buts (Sophos)



© Copyright Cognisys Consultants Inc., 1989, 1990.

Figure 8 : Arbre d'inférence construit par Sophos

Pour utiliser Sophos, il faut avoir au préalable une bonne perception des concepts et règles qui décrivent le domaine à élucider. Ce type d'expertise dite à théorie forte signifie que les

attributs caractérisant chaque but et sous-but sont connus, de même que les corrélations qui existent entre eux et leurs valeurs possibles; le logiciel permet à l'expert d'expliciter les facteurs clefs qu'il utilise pour résoudre son but. Nous verrons plus loin qu'il existe des domaines où cette théorie forte n'existe pas (c'est le cas d'une expertise à théorie faible), et où Sophos est indéniablement inefficace.

Sophos fonctionne de façon interactive avec un expert (et l'aide éventuelle d'un cogniticien) qui fournit au logiciel les sous-buts et leurs valeurs possibles; l'expert doit les connaître, l'expertise étant à théorie forte. À partir de ces sous-buts et de leurs valeurs possibles, le logiciel demande systématiquement à l'expert de fournir la valeur du but pour chaque combinaison possible des valeurs des sous-buts : "Aux valeurs des sous-buts x, y, et z, correspond quelle valeur du but à atteindre ?" Une explosion combinatoire est à anticiper si le domaine d'expertise est vaste et comporte un grand nombre de sous-buts. De plus, la découverte de la base de connaissances est très monotone et le logiciel ne demande pas à l'expert s'il est certain de n'avoir rien oublié, donc l'élucidation est peu fiable. D'ailleurs, durant cette phase, les touches les plus utilisées sont F6 et F7 (sur le clavier IBM), rendant le logiciel très monotone d'utilisation.

Au terme de cette phase de découverte des connaissances, Sophos est en mesure d'exporter les règles ainsi générées. En cours d'utilisation, Sophos n'a qu'une façon d'illustrer sa base de connaissances, soit par l'arborescence des sous-buts. Toutefois, le peu d'information disponible sur celle-ci la rend à peu près inutile (voir figure 8). En effet, il est impossible de voir les valeurs que peut prendre chaque nœud d'inférence, ce qui serait probablement utile pour la maintenance, ou même l'optimisation.

Au niveau de la convivialité, Sophos est très facile à apprendre, mais la mise à jour de la base de connaissances n'est pas aisée, ayant à naviguer dans un arbre qu'on n'a pas toujours sous les yeux (voir figure 9). Sa documentation ne dépasse pas les limites du logiciel, mais est claire et bien ordonnée. L'interface à l'utilisateur semble faible; nous avons noté une surutilisation de touches de contrôle mal situées, l'emploi de touches alternatives (ALT) pas toujours indiqué, etc. C'est donc un logiciel qui nous semble peu stimulant sur le plan de la convivialité.

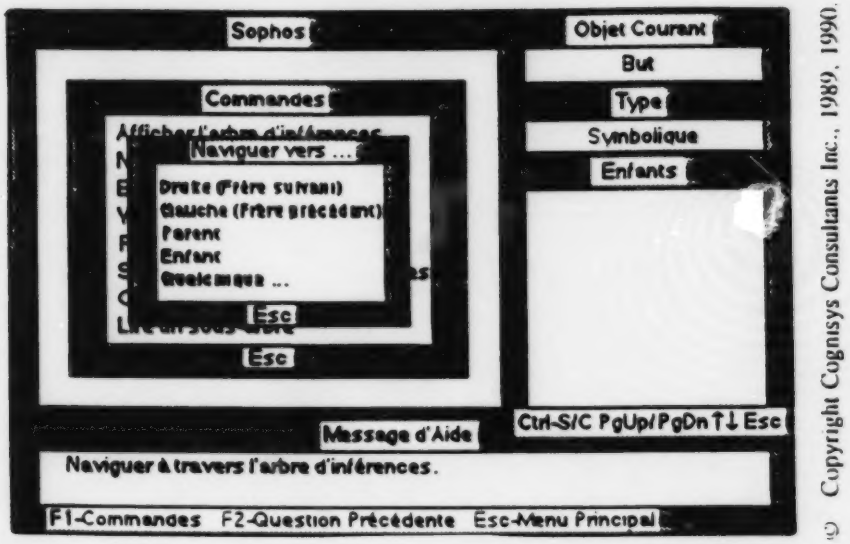


Figure 9 : Navigation dans l'arbre (Sophos)

2.1.2 KAT

KAT (Knowledge Analysis Tool) a été développé et est commercialisé par un cogniticien, Gary S. Ribar [13]. Ce LAC fonctionne sur Macintosh et coûte 395 $ US. Le fournisseur n'offre pas de service pour son produit. Le logiciel n'est pas limité à un domaine d'application spécifique. Une macro construisant une base de connaissances sommaire sur les prêts bancaires est fournie avec le logiciel.

Ce logiciel est basé sur une architecture séparée et est conçu spécifiquement pour développer des bases de connaissances pour le GSE Level5; seul Level5 peut donc exploiter la base de connaissances produite.

KAT permet à l'expert ou au cogniticien de créer une arborescence graphique décrivant l'ensemble du processus décisionnel de l'expert. Il produit des règles à partir de cet arbre de décision où la racine est le but à atteindre (appelé ici conclusion), et les nœuds sont les attributs qui déterminent ce but (voir figure 10). KAT permet de structurer les buts, les sous-buts et les attributs; l'utilisateur crée une description en "chaînage arrière" des attributs

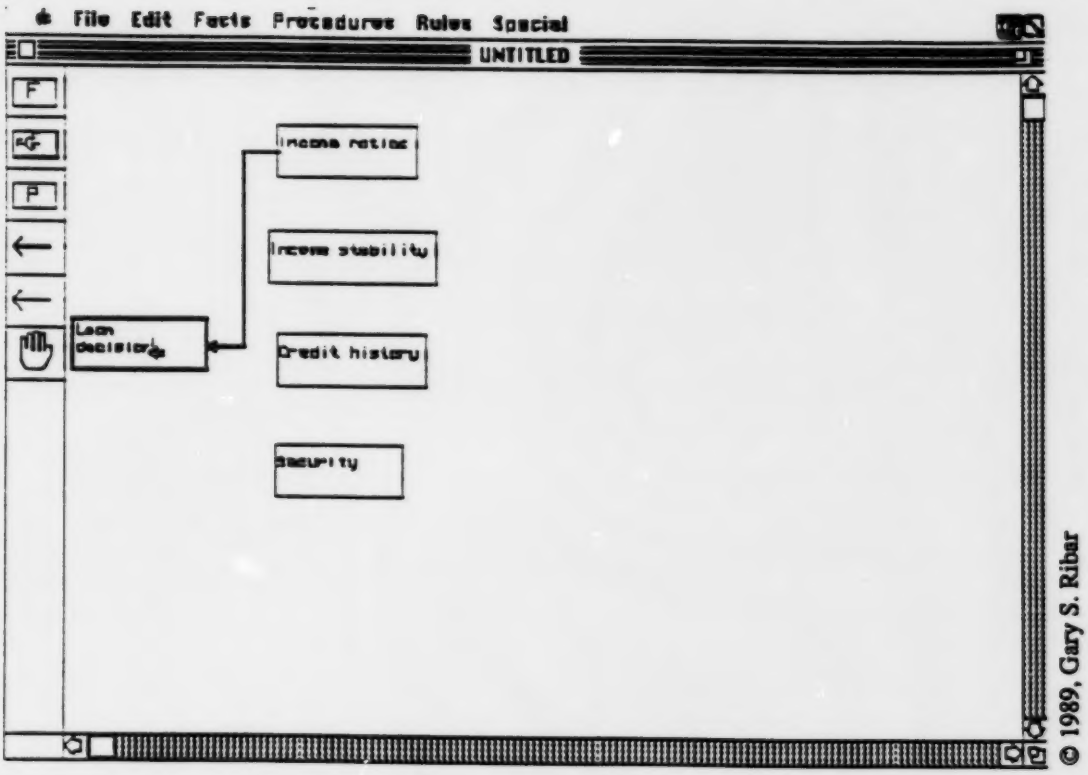


Figure 10 : Arbre d'attributs relatifs à un but donné (KAT)

impliqués dans le processus de décision, et le tout graphiquement. Ainsi, KAT traite très bien les problèmes de classification.

De plus, KAT peut accepter dans sa zone de travail plusieurs hypothèses concourantes, et les passages d'une hypothèse à l'autre sont possibles (c.-à-d. les règles générées testent des hypothèses dans un ordre défini par l'expert), rendant des problèmes de diagnostic possibles (voir figure 11); ces "go to" ne sont pas très élégants, mais cette décomposition du processus de décision correspond plus adéquatement aux problèmes de diagnostic qu'un arbre de décision unique (NEXTRA propose une autre approche intéressante).

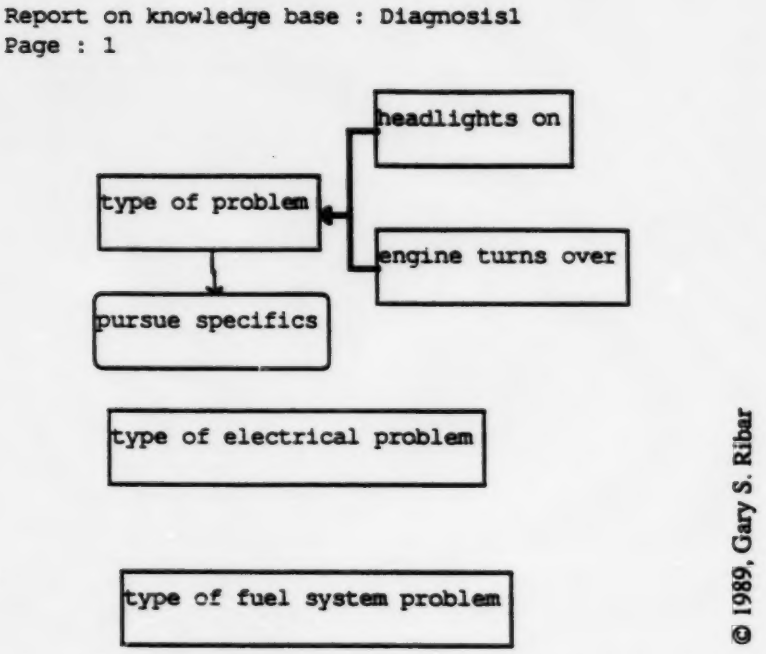


Figure 11 : Hypothèses diverses participant à une base de connaissances (KAT)

KAT nécessite une expertise à théorie forte. En effet, et tout comme Sophos, si le domaine n'est pas a priori formalisé, il est inutile de commencer à travailler avec ce logiciel, car celui-ci n'a pas pour fonction de trouver les relations existantes entre les informations; il présuppose que l'expert les connaît déjà et peut les fournir dans le format de représentation du LAC.

KAT fonctionne en interactif où doivent lui être fournis les sous-buts et leurs relations avec le but. L'arbre (ou les arbres) de décision s'élabore entièrement à l'intérieur d'une page graphique grâce à des utilitaires contrôlés par la souris. Pour définir un facteur, il suffit de le sélectionner. En fait, l'utilisateur est entièrement libre de procéder dans l'ordre qu'il désire. À partir d'un arbre de décision, KAT va combiner toutes les valeurs possibles des

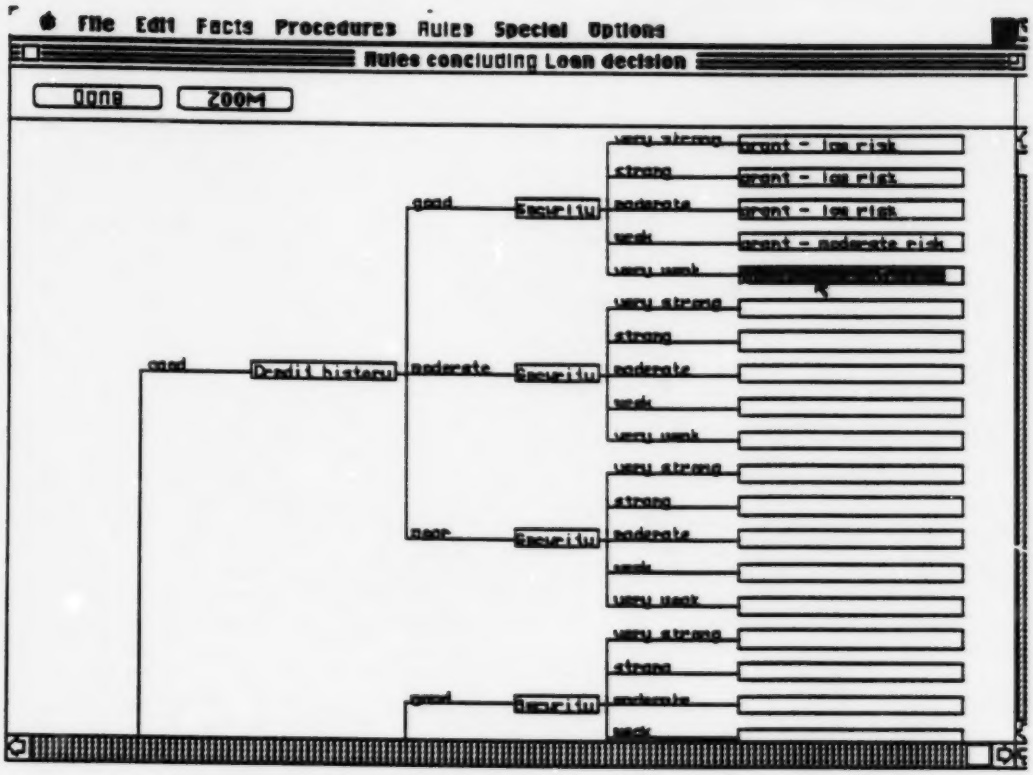


Figure 12 : Conclusion fournie interactivement par l'expert pour une combinaison d'attributs (KAT)

facteurs et demandera pour chaque combinaison le résultat de la conclusion (voir figure 12).

L'approche est similaire à celle décrite précédemment pour Sophos. Toutefois, si le logiciel détecte, pour toute une branche de l'arbre, que les facteurs feuilles n'ont aucune incidence sur la valeur de la conclusion (c.-à-d. n'aident pas à discriminer), il va élaguer l'arbre, réduisant d'autant le nombre de règles produites, qui autrement auraient été redondantes et auraient encombré inutilement le SE.

En sortie, KAT peut produire un rapport complet et bien présenté de la base de connaissances, avec arbre de décision et arbre des valeurs possibles à l'appui, ce qui accélère grandement la mise au point. Aussi, il peut exporter ses connaissances dans un

fichier en format Level 5 immédiatement utilisable. Une des forces de KAT (mais en même temps une faiblesse) est qu'il est conçu très spécifiquement pour Level 5. Avec KAT, il est facile d'ajouter des opérandes et des accès à des bases de données conventionnelles que pourra interpréter Level 5, donnant ainsi une base de connaissances plus autonome et plus complète.

KAT exige au plus une heure d'apprentissage de la part de l'utilisateur. L'utilisation est évidente et efficace grâce aux icônes et autres conventions du Macintosh, et la mise au point d'une base de connaissances ne demande pas de connaître des touches de contrôles sophistiquées. Ceci est rendu possible grâce à l'interface respectant la philosophie multi-fenêtre. Finalement, la documentation est sans défaut.

2.1.3 CAMEO

Le plus générique des logiciels évalués est CAMEO (Computer-Aided Modeling of Expertise in Organizations), développé par Arthur D. Little [4]. Non-commercialisé, le logiciel est accessible aux clients de la firme ADL. Il fonctionne sur Macintosh (Plus, SE, II) et est écrit en HyperTalk, le langage de HyperCard. Le résultat peut être traduit automatiquement en code utilisable par KEE et Nexpert *Object* (ADL peut aussi exporter vers d'autres coquilles). Le résultat peut même servir à initialiser un réseau de neurones (modèle connexioniste) exportable vers les logiciels Macbrain et Neuralworks. Par sa puissance graphique et sa modularité, CAMEO est conçu spécifiquement pour permettre à plusieurs développeurs de travailler sur un même projet.

CAMEO permet aussi l'analyse de transcriptions (entrevues avec les experts, documentation, manuels). Il peut lire un fichier ASCII dans une pile HyperCard. L'utilisateur peut aisément localiser et lier du texte aux règles et au modèle de connaissances pour fins de documentation.

Les informations sont représentées sur des cartes dont les séquences forment des piles; les cartes sont liées entre elles pour former un graphe d'inférence. Chaque carte contient des champs et des boutons à cliquer pour effectuer les diverses opérations. La navigation parmi les cartes est simple (carte par carte), mais des utilitaires permettent aussi de naviguer beaucoup plus efficacement.

Un modèle de connaissances comporte un modèle de données (pour structurer les données et les informations de l'application) et, si désiré, un modèle de processus (pour structurer les activités, fonctions et stades de l'application); les deux modèles sont intégrés via les règles de décision d'un réseau d'inférences. Les éléments du modèle de connaissances sont des objets, leurs attributs, les assertions relatives à certaines valeurs d'attributs et les règles. Avec en plus un modèle de processus, des éléments de type processus et décision sont ajoutés. Chaque élément correspond à une carte d'une pile.

La figure 13 décrit un tel modèle de données : l'objet *Insurance Application* possède d'autres objets (*Proposed Insured*) et a des propriétés décrites par les attributs (*Coverage Amount*), ayant elles-mêmes des valeurs décrites par les assertions (< *$10K*). Les éléments sont donc reliés verticalement en une hiérarchie d'appartenance des objets conceptuels importants pour l'application. Les éléments de type règle combinent les assertions pour créer le graphe d'inférence; les flèches A et B représentent deux règles de décision.

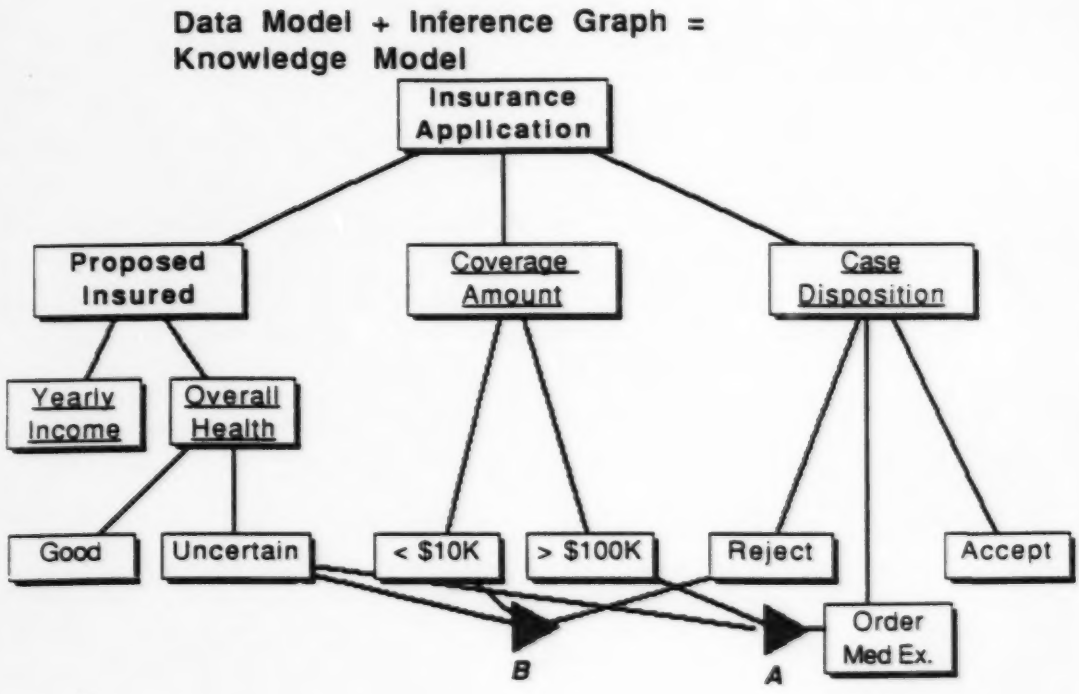


Figure 13 : Un modèle de données dans le domaine des assurances (CAMEO)

La figure 14 décrit un modèle de processus, similaire au précédent; un tel modèle fournit une autre façon d'organiser les processus et les éléments de type règles. L'élément de type processus *Underwriting* est composé d'autres processus (sous-processus comme *Assess Medical*) et implique des éléments de type décision (*Overhall health*) qui, à leur tour, utilisent des éléments de type règle.

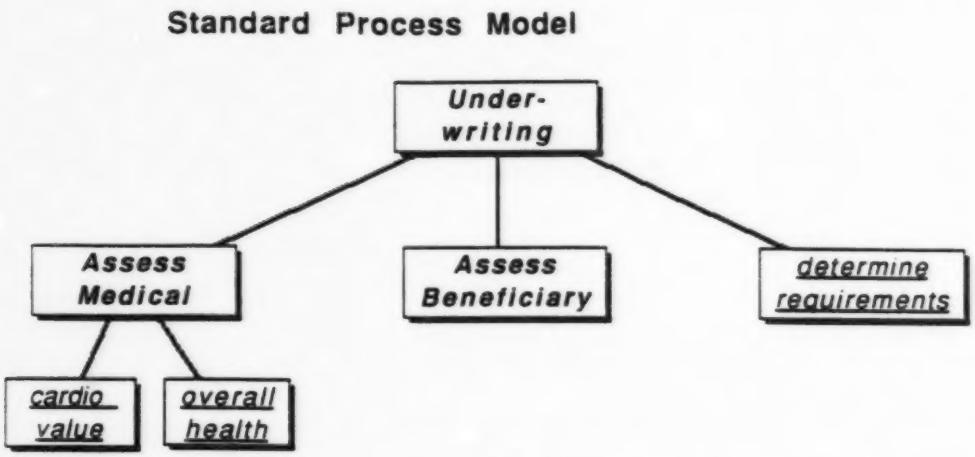


Figure 14 : Un modèle de processus dans le domaine des assurances (CAMEO)

L'édition de la base de connaissances est très aisée. Une vue graphique locale ou globale (voir figure 15) des éléments et des relations entre eux est toujours disponible et permet, en cliquant sur les éléments, de naviguer rapidement. Cette vue graphique peut être obtenue selon divers niveaux d'abstraction plus ou moins élevés, en fonction du niveau de détail désiré par l'utilisateur.

La figure 16 illustre une carte pour un élément de type règle avec les boutons et les champs correspondants; il est facile de passer à la carte des prémisses, des conclusions, du parent, des enfants (selon les relations définies précédemment), accéder au menu des commandes, voir le champ de documentation de l'élément. Par exemple, à partir d'une carte de type assertion, l'utilisateur peut rapidement identifier toutes les règles qui affectent ou sont affectées par cette assertion. La mise à jour du modèle est donc grandement facilitée par la puissance graphique du logiciel et par la modularité des informations à manipuler.

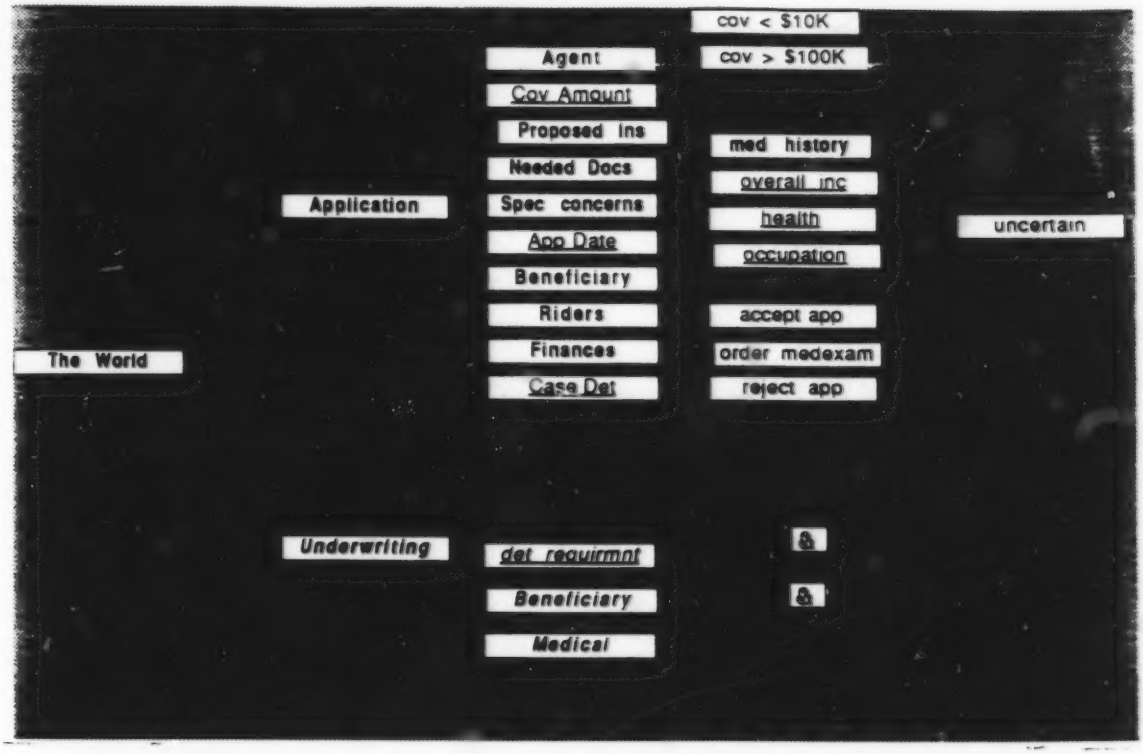


Figure 15 : Une vue globale des éléments d'un modèle de connaissances (CAMEO)

CAMEO est basé sur une architecture séparée. Il est le plus générique des logiciels tout en étant vraiment un outil conceptuel pour supporter les premières phases de modélisation des connaissances. Il est donc utile pour développer l'application par la suite, quelle que soit la coquille de développement de SE choisie. Il se situe au-delà des arbres de décision décrits précédemment, ces derniers n'offrant pas les mêmes possibilités ni la même flexibilité. Il permet de traiter autant les problèmes de classification que de diagnostic. Il nécessite toutefois une expertise à théorie forte, l'utilisateur (probablement cogniticien) étant entièrement responsable d'identifier les éléments qu'il doit organiser dans ses piles de cartes, et de les structurer convenablement.

Basé sur la philosophie HyperCard, CAMEO est donc très facile à apprendre et agréable à utiliser; l'utilisateur élabore rapidement son modèle en cliquant sur les boutons ou sur les éléments qui composent le graphe d'une vue locale ou globale, ou en appelant les menus. Enfin, la documentation fournie est claire et complète.

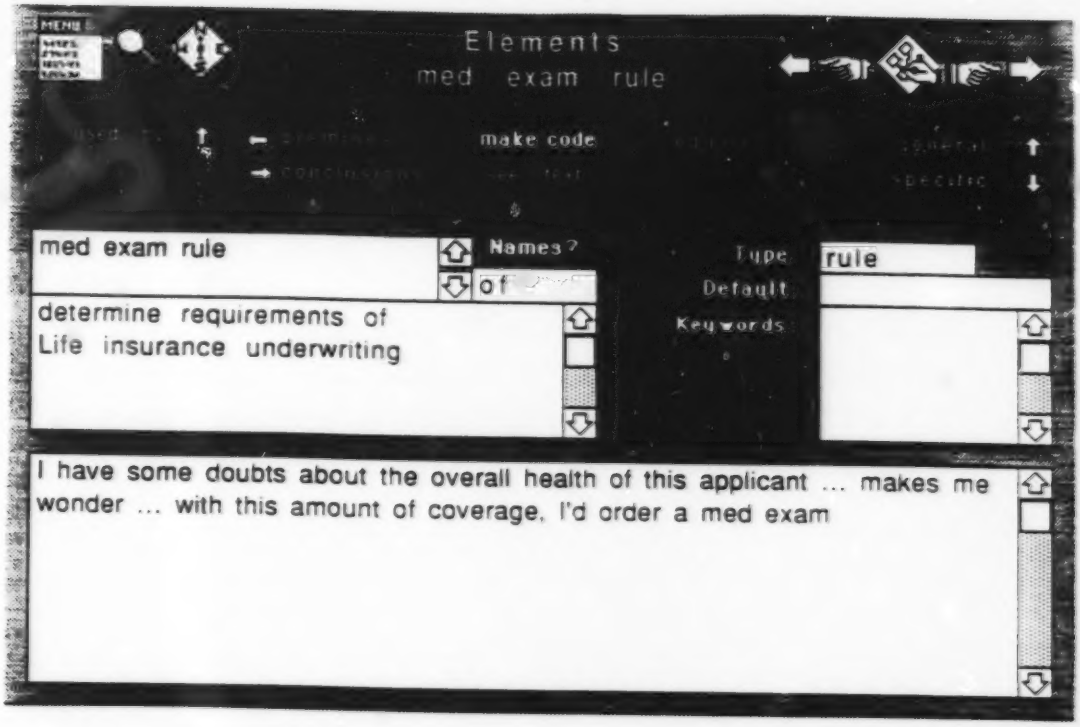


Figure 16 : Elément de type règle représenté par une carte HyperCard (CAMEO)

Toutefois, nous émettons des réserves quant à son potentiel pour de très vastes projets et pour certaines catégories de développeurs. Le fait qu'il représente chaque élément sous forme d'une carte reliée aux autres par des relations peut causer une certaine lourdeur dans la représentation. Par exemple, chaque assertion décrivant une valeur ou un intervalle de valeurs important est représentée par un nœud dans le graphe, avec un grand nombre de connexions pour toutes les relations qui impliquent cette assertion; le nombre d'assertions à manipuler peut devenir très important. De plus, l'utilisateur ou le développeur qui veut exploiter une représentation de type objet (avec ses liens d'héritage, ses méthodes et ses attributs) ne trouve pas en ce logiciel une approche similaire. Cependant, les utilisateurs non spécialisés ou n'ayant pas adopté une telle méthodologie de développement trouveront sans doute en CAMEO un potentiel véritable.

2.1.4 IXL

IXL est un logiciel produit par Intelligence Software [12] et vendu au prix de 500 $ US. Le fournisseur ne semble pas fournir de service après-vente. IXL fonctionne sur les micro-ordinateurs IBM, et travaille à partir de fichiers de données en format DBF (fichier DBase), DIF (Lotus), ou simplement en format ASCII. Ce LAC permet d'extraire des connaissances directement à partir de données existantes, organisées dans une base de données, et sans l'intervention d'un cogniticien.

Le logiciel IXL est à architecture séparée. En entrée, il reçoit un fichier de données sous la forme d'un tableau. Chacun des champs de ce tableau est un but potentiel pour la découverte de règles. Plus exactement, sa sortie est un ensemble de règles ayant comme conclusion une expression de la forme "*champ* = *valeur*" où *champ* est un champ de ce tableau et *valeur* l'une de ses valeurs possibles. De plus, ces règles peuvent être exactes ou

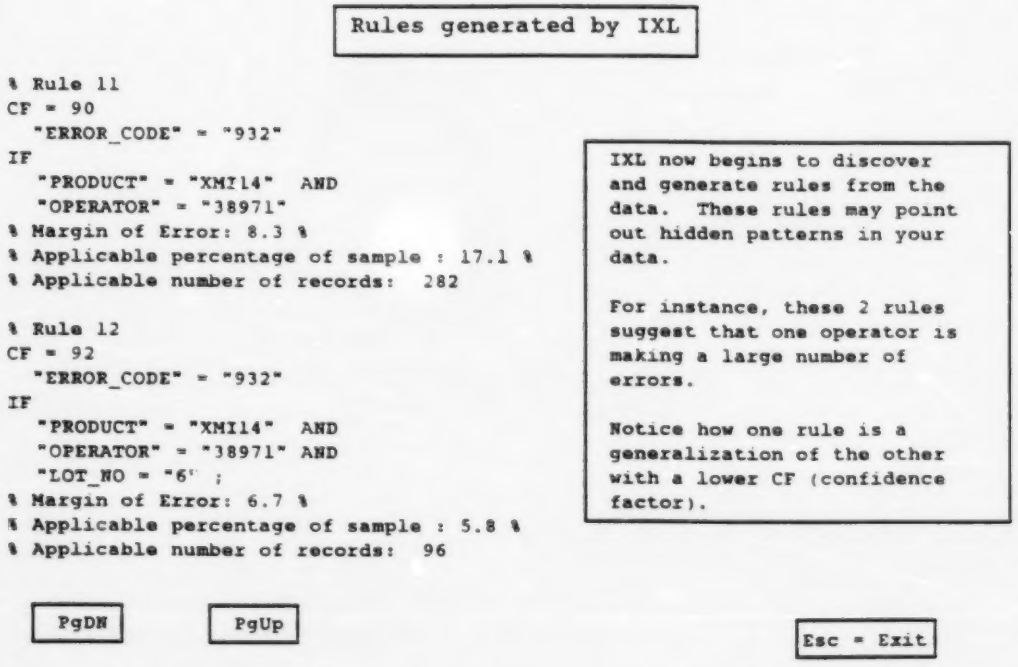


Figure 17 : Deux règles générées par IXL à partir d'une table d'enregistrements.

inexactes selon un facteur de certitude choisi par l'utilisateur ou fixé à 80% par défaut. Malheureusement, il ne trace aucun diagramme ou réseau des règles obtenues, ce qui, selon nous, aiderait l'utilisateur à visualiser les relations. Le logiciel produit une sortie en format texte respectant au choix la norme Prolog, Lisp, Guru, Intelligence Compiler ou Personal Consultant Plus.

La figure 17 illustre deux règles générées par IXL à partir d'une table de quelques centaines d'enregistrements. Avec une certitude de 80%, le champ *code d'erreur* prend la valeur *932* pour un certain produit avec un certain opérateur; une règle plus spécifique, avec un facteur de certitude de 92%, spécifie le numéro de lot en plus du produit et de l'opérateur.

Ce logiciel n'est donc pas destiné à l'aide à l'acquisition des connaissances d'un expert, mais plutôt à la découverte de règles implicitement contenues dans un tableau de données. Ces règles peuvent par la suite constituer une base de connaissances en soi, mais la publicité du produit fait plutôt mention d'un gain immédiat par la simple connaissance des règles découvertes. Cette dernière façon d'utiliser le logiciel semble être plus en accord avec la structure des règles construites.

Les interrogations sont simples et directes, mais malheureusement, quelque peu abstraites : l'utilisateur doit spécifier le niveau de confiance (facteur de certitude), de signification, de généralité et la marge d'erreur désirés avant que le logiciel ne procède à une recherche d'inférence.

Il y a toutefois une ombre au tableau : les performances temporelles ne semblent pas sensationnelles. Nous ne disposions certes que d'une disquette de démonstration pour les fins de notre évaluation, mais le fournisseur mentionne qu'il faut sept minutes pour rechercher les inférences dans un fichier de 1 600 enregistrements de cinq champs. À ce rythme, et même avec un algorithme de l'ordre de "n log n", il faudrait cinq heures pour traiter 50 000 enregistrements.

Par contre, IXL est très facile à utiliser; l'interaction se déroule par des fenêtres, avec des valeurs de défaut préfixées. Nous émettons toutefois certains doutes sur la qualité de ce produit dont la publicité "technique" demeure vague sur les méthodes employées pour la génération des règles; nous supposons que des techniques d'inférence statistique (tests d'hypothèses et d'ajustement) sont à la base de la méthode d'induction. Nous n'avons pas eu accès à la documentation et n'avons pas été en mesure d'éprouver le logiciel avec nos propres données; il nous est donc impossible d'émettre un jugement plus précis.

2.1.5 NEXTRA

NEXTRA est un logiciel commercialisé par Neuron Data [21] au coût de 4 000 $ US. Il est disponible sur de nombreux ordinateurs, allant du IBM AT et du Macintosh jusqu'au Vax en passant par les Sun 3. Il est basé sur une architecture séparée et peut être utilisé pour la classification et le diagnostic. La base de connaissances résultante doit être exportée vers Nexpert *Object* pour être mise en œuvre.

Le logiciel est basé sur plusieurs études théoriques publiées récemment et dont les origines remontent aux travaux de George Kelly des années 1950 [14]. Les grilles de répertoire sont utilisées pour identifier les traits et les attributs et pour décrire les objets en termes de ces traits et attributs [22]. En voici les idées générales :

Un domaine d'expertise comporte généralement un ensemble d'entités différenciées entre elles par des attributs appropriés. Par exemple, un informaticien peut distinguer les langages Basic et Pascal en fonction d'attributs comme *procédure paramétrisée*, *syntaxe libre*, *définition de type*, *convivial*. L'expertise permet de graduer la valeur relative d'un attribut pour chacune des entités. En pratique, un ensemble de paires d'attributs opposés sont utilisés, comme *syntaxe libre* et *syntaxe limitée*, et une échelle graduée de 1 à 9 pondère la présence d'un attribut ou de son opposé. Une entité est alors caractérisée par les valeurs entre 1 et 9 de chacun des attributs identifiés.

À l'aide de NEXTRA, un domaine d'expertise est décrit en donnant une liste d'entités et de paires d'attributs; ces deux listes sont indépendantes. Aucun sens n'est rattaché aux attributs et aux entités, et ces noms peuvent être entrés dans un ordre quelconque. Une interface graphique permet de décrire chaque entité en donnant une valeur pour chacune des échelles formées par les paires d'attributs.

Le mode d'entrée des entités et des attributs est très plaisant, le logiciel ne précipitant jamais l'utilisateur-expert. Dans certains cas, le logiciel tentera d'obtenir de l'expert des entités ou des attributs supplémentaires en lui posant des questions comme : "Les entités X et Y sont très semblables. Pouvez-vous fournir un nouvel attribut qui les distinguerait ?" (voir figure 18). Grâce à ces nombreux questionnements, nous obtenons une description de plus en plus fine des entités en fonction des attributs, et une différenciation de plus en plus nette entre les entités.

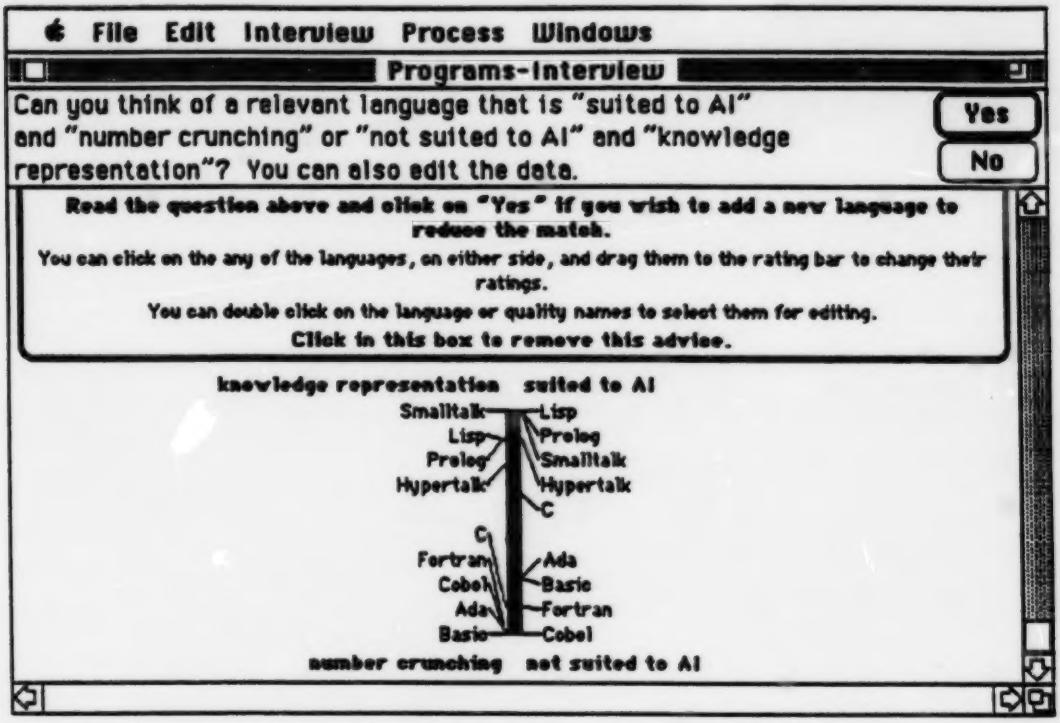


Figure 18 : Disposition des entités sur deux échelles de graduation des attributs (NEXTRA).

NEXTRA est facile à utiliser grâce à ses conseils disséminés dans les fenêtres. Pour mettre au point une base de connaissances, rien n'est plus aisé que d'entrer de nouvelles entités. Tout cela est géré par une interface de qualité irréprochable.

NEXTRA peut présenter la base de connaissances de trois façons en utilisant au maximum ses possibilités graphiques, ce qui permet d'avoir une bonne vision de la base de connaissances en cours de conception : par matrice entités/attributs, par représentation spatiale et par groupes apparentés.

L'affichage sous forme matricielle est une simple reproduction de toutes les entités avec les valeurs des attributs (voir figure 5). Il fournit un moyen direct de voir l'ensemble des connaissances.

L'affichage spatial utilise une méthode développée par Slater [25], et basée sur l'analyse en composantes principales *(principal component analysis)*. Chaque entité est considérée comme un point dans un espace multidimensionnel dont le nombre de dimensions est le

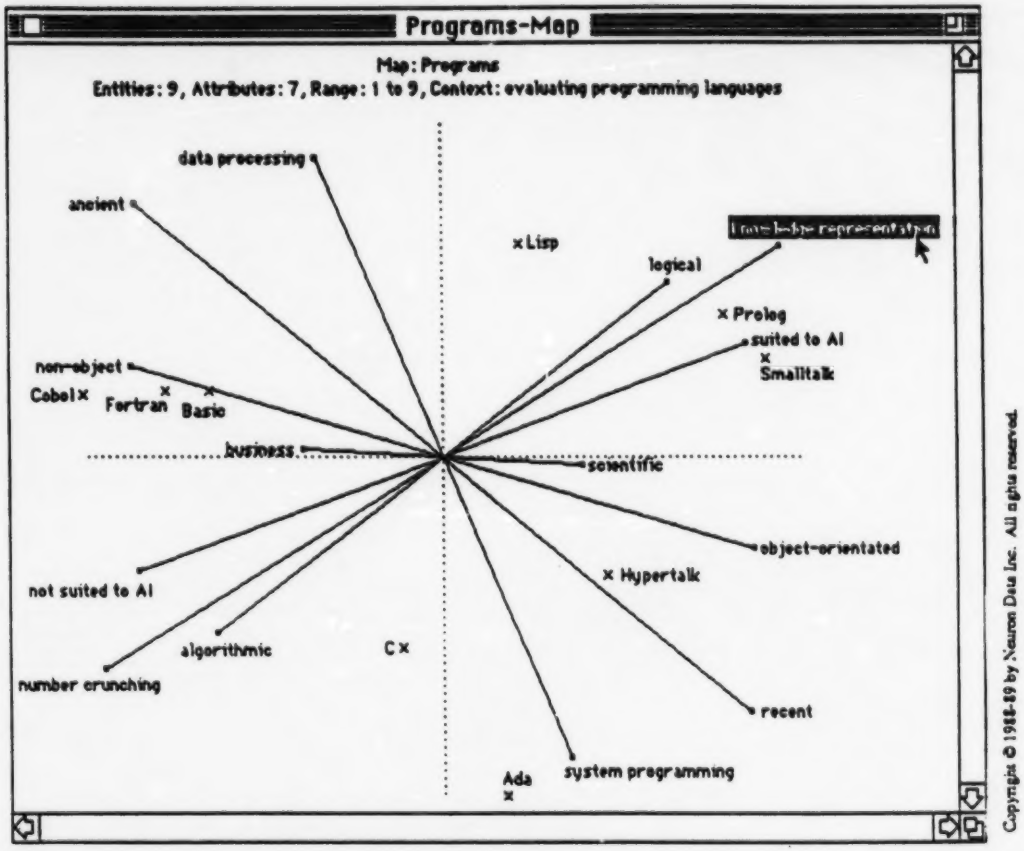


Figure 19 : Représentation spatiale projetée sur un espace bidimensionnel (NEXTRA).

nombre d'attributs. La distance entre deux points définissant deux entités exprime un degré de similarité entre celles-ci. Pour visualiser cet espace en deux dimensions, une projection est effectuée sur deux axes principaux déterminés par le logiciel pour mieux faire ressortir les différences et les similitudes entre entités : l'axe horizontal exprime les attributs qui décrivent la distance maximale entre les entités et l'axe vertical exprime les attributs décrivant la seconde distance maximale (voir figure 19). Pour une plus grande précision, il est possible d'avoir un affichage numérique des composantes des points dans l'espace.

L'affichage par groupes apparentés repose aussi sur la notion d'espace multidimensionnel, mais les distances entre entités sont utilisées pour les regrouper deux à deux; deux entités ayant la plus courte distance entre elles sont reliées. Après un premier regroupement, le même processus est répété entre les paires d'entités et les entités non regroupées et ainsi de

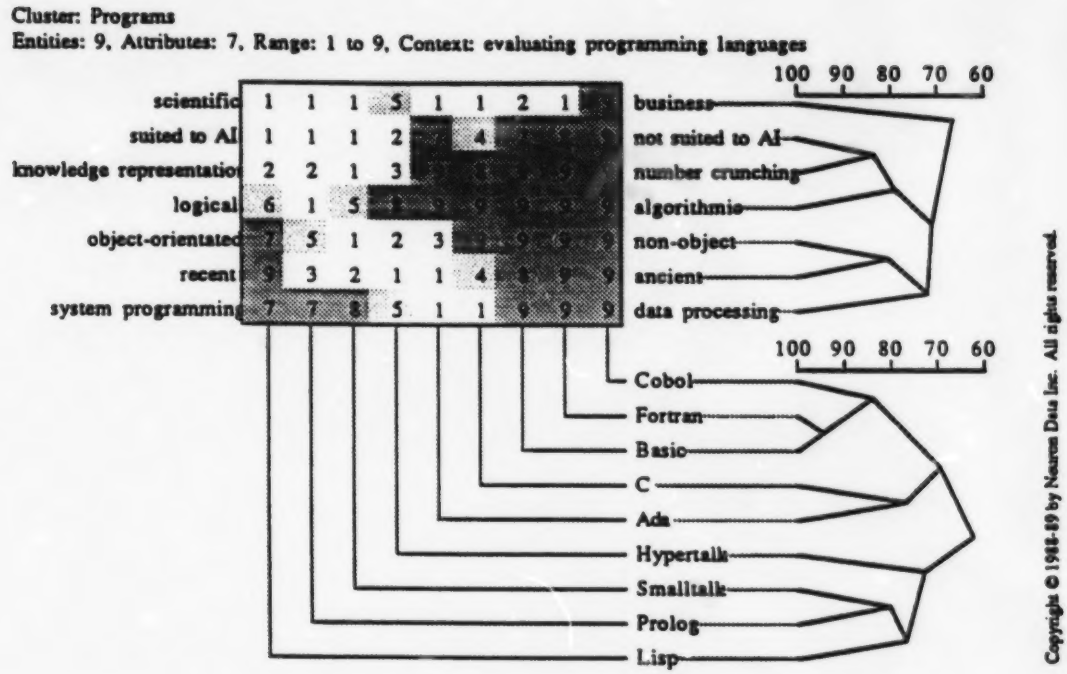


Figure 20 : Représentation par analyse des regroupements (NEXTRA).

suite en une hiérarchie : la partie droite de la figure 20 représente cette hiérarchie avec une graduation en pourcentage du degré de similarité. La même procédure de regroupement est appliquée aux attributs. Le tableau ordonne ainsi les entités et les attributs par leur degré de similarité. Cette représentation est fondée sur les travaux de Gaines & Shaw dans le domaine de l'analyse des regroupements (*cluster analysis*) [24].

Il est possible de comparer deux ensembles d'entités/attributs ayant les mêmes entités ou attributs. Cette méthode permet de comparer les opinions de deux experts afin d'établir un consensus entre eux dans un domaine donné. La méthode est partiellement fondée sur les travaux de B.R. Gaines et M.L. Shaw [9]. La sortie numérique décrit, par une liste de paramètres, les différences et similitudes entre les bases de connaissances.

Deux modalités de transfert des bases de connaissances vers des GSE sont disponibles. La première permet une sortie textuelle indépendante du GSE et l'autre crée une base de connaissances directement utilisable par Nexpert *Object*. Notons que la première méthode

peut être considérée comme une façon alternative de visualiser la base de connaissances. Pour Nexpert *Object*, non seulement des règles sont produites, mais des classes et des objets sont automatiquement définis. Les règles sont générées avec des cœfficients de priorité pour diriger la consultation du SE de telle sorte que les conclusions les plus probables soient testées d'abord.

La documentation est à la fois complète, expliquant même le concept de fenêtre, et intéressante particulièrement à cause des exemples sélectionnés. Toutefois, la signification des résultats numériques produits par certains algorithmes, par exemple lors de la comparaison entre les ensembles d'entités/attributs, nous semble difficilement compréhensible; il peut s'avérer nécessaire de consulter les références scientifiques correspondantes.

2.1.6 Cogensys Judgment Software

Cogensys Judgment Software est un produit de la firme Cogensys [7], et est vendu au prix de 250 000 $ US, installation et service inclus; ce prix comporte toutefois des services de consultation, à savoir le développement sur mesure et l'installation de l'application requise par le client. Le logiciel est supporté par les ordinateurs IBM PC et Macintosh. Il peut servir dans des domaines d'applications très généraux; il a déjà été installé et utilisé avec succès sur 11 sites, dont une compagnie d'assurance à Toronto.

Avertissement : comme nous n'avons pu avoir un exemplaire du logiciel à notre disposition, nous ne pouvons baser notre évaluation que sur la présentation du logiciel faite par Cogensys au CRIM, en mars 1990, et sur les quelques références fournies. Nous ne sommes donc pas en mesure de fournir une analyse détaillée de ce logiciel.

L'architecture de Cogensys Judgment Software est intégrée, c.-à-d. que le logiciel effectue autant l'acquisition des bases de connaissances que leur mise en œuvre une fois qu'elles sont suffisamment élucidées.

Ce LAC apprend à partir de cas, ou exemples résolus, dont il sélectionne et conserve les plus instructifs. À partir d'une banque de ces cas, il est en mesure d'inférer la solution d'un nouveau cas proposé en entrée.

Toutefois, le logiciel semble avoir certaines lacunes au niveau de l'organisation de sa banque de cas. Citons, à cet effet, le *Cogensys Judgment Processor Technical Overview* [7] : "For each situation in the judgment base, an assertion level is generated which indicates how logically relevant the situation is to the example presented". Selon notre interprétation, le parcours de la banque de cas résolus semble inefficace surtout lorsque celle-ci est vaste. Dans l'une des références fournies, il est suggéré, à l'intérieur d'un réseau, de réserver un serveur uniquement pour le logiciel afin de ne pas trop encombrer l'ordinateur principal.

La documentation technique laissée à notre disposition ne précise pas les algorithmes d'apprentissage. Toutefois, nous avons appris que le système utilise, dans un espace topologique de dimension *n*, des vecteurs de *n* attributs qui correspondent aux cas fournis. Lorsqu'un nouveau cas lui est présenté, il recherche dans sa base un cas résolu qui lui correspond exactement; s'il n'en trouve pas, il trouve le vecteur (cas résolu) dont la "distance" au nouveau cas est minimale et fournit à l'utilisateur le même diagnostic. La mesure de "distance" pour déterminer dans la base le cas le plus similaire au nouveau cas dépend des attributs que l'expert aura identifié comme prédominants : la proximité d'un vecteur à un autre dépend de l'importance relative des attributs entre eux. En effet, le logiciel construit un arbre décisionnel où l'importance relative des attributs (poids) peut être fournie par l'expert ou inférée par le système selon une procédure qui n'est pas révélée dans la documentation.

Grâce à un logiciel d'appoint, le *Information Manager*, le *Judgment Processor* est en mesure de recevoir des cas autant d'une base de données qu'en mode interactif, facilitant ainsi la période d'apprentissage. Toute la manipulation des données se fait à partir de formulaires à l'écran; il n'y a donc aucune illustration graphique. Les interrogations sont minimales, mais remplissent bien leur fonction.

Grâce aux autres logiciels, *Application Manager* et *Information Manager*, Cogensys semble fournir un environnement de prise de décision adapté à l'entreprise privée. Cependant, nous ne pouvons nous prononcer sur leur facilité d'apprentissage et d'utilisation. Les interfaces semblent puissantes et facilement adaptables en réseau.

Le logiciel Cogensys Judgment Software se distingue des autres LAC faisant l'objet de cette étude en ce sens qu'il propose davantage des services de consultation (développement complet d'une application) qu'un simple logiciel à être exploité par l'expert.

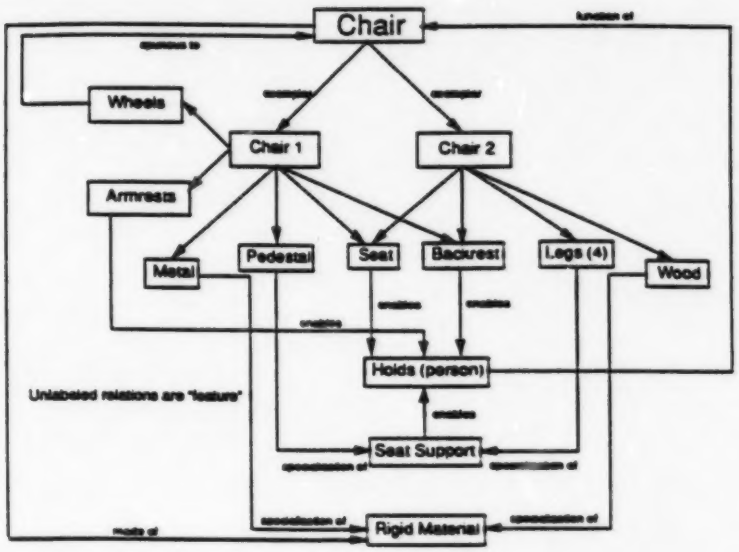
2.1.7 CL-Protos

Le dernier LAC que nous avons évalué, CL-Protos, est un logiciel issu de recherches sur l'acquisition des connaissances réalisées aux universités de Austin et de Vanderbilt [6]. Il représente une approche innovatrice à l'apprentissage de concepts dans le processus de classification heuristique. Il n'est pas commercialisé, mais les auteurs acceptent d'en diffuser gracieusement des exemplaires. Ce logiciel met en application les concepts développés par E. Ray Bareiss et Bruce W. Porter dans leurs travaux de recherche [2] [3]. CL-Protos est écrit en Common Lisp et fonctionne sur plusieurs plates-formes, dont Sun, HP 9000 et les Macintosh. Sous sa forme actuelle, le logiciel ne permet toutefois pas d'exploiter automatiquement ses connaissances et de les exporter vers d'autres logiciels.

CL-Protos réalise l'acquisition de connaissances pour la classification par l'apprentissage par analogie. Son apprentissage peut se faire de deux façons : par la présentation d'un nouveau cas ou par l'explication de la relation entre un nouveau cas et un cas existant. En fait, l'utilisateur (ici l'expert) présente un nouveau cas, puis CL-Protos le classifie par concordance (*match*) avec des exemples similaires et explique sa classification en exposant la concordance découverte. CL-Protos améliore ensuite sa classification en questionnant l'expert pour acquérir des connaissances d'appoint, puis il enregistre ce nouveau cas comme un nouvel exemple.

L'utilisateur fournit un nouveau cas sous forme d'un ensemble de caractéristiques descriptives. Par exemple, pour décrire *automobile*, l'utilisateur peut entrer l'ensemble *{(corps métal), radio, pneus, sièges, (roue 4), moteur}*; une paire comme *(corps métal)* représente une caractéristique à l'aide d'un attribut *corps* et de sa valeur *métal*. Le nombre de caractéristiques est quelconque et celles-ci peuvent être des symboles, comme *moteur*, ou des listes générales.

CL-Protos traite un nouveau cas en établissant un degré de similitude avec un cas existant, puis en transformant adéquatement la solution du cas existant pour qu'elle corresponde au nouveau cas. La figure 21 (partie supérieure) illustre le concept de *chair* représenté par deux exemples *chair1* (en métal sur roulettes) et *chair2* (en bois sur quatre pattes). La classification d'un nouveau cas implique la recherche d'un exemple qui concorde fortement avec ce cas. Cette recherche utilise deux types de connaissances : les relations entre concepts (par exemple, le fait qu'un siège permet de supporter une personne) et les




```
CHAIR1 ( a term/exemplar )
  Category:     Chairs, typicality 0.80
  Features:     (wheels, armrests, metal, pedestal, seat, backrest)
  Diff links:   ((legs 4)) --> chair2
  Created by:   Dvorak

CHAIRS  ( a term )
  Exemplars:    (chair1, chair2)
  Importances:  seat 0.70, backrest 0.60, armrests 0.70, pedestal 0.30,
                (legs 4) 0.30, metal 0.00
  Relations:    chairs has part metal, strength=0.70
                chairs has part wood, strength=0.70
                chairs has function (holds person), strenght=1.00
  Created by:   Dvorak

(holds person)          ( a term )
  Relations:    (holds person) coocurs with lateral support,
                strenght=0.80
                (holds person) is enabled by seat, strenght=0.90
                (holds person) is enabled by backrest, strenght=0.90
                (holds.person) is enabled by seat-support, strenght-0.90
                (holds person) is function of chairs, strenght=1.00
  Created by:   Dvorak
```

Figure 21 : Apprentissage de concepts pour la classification basé sur des exemples (CL-Protos)

importances relatives des caractéristiques (par exemple, *wheels* est optionnelle mais *seat* est essentielle à la catégorie *chair*).

Pour une base de connaissances ayant à discriminer une quinzaine d'attributs, il faut, selon M. Van der Schalie, représentant de Cogensys, un millier de cas résolus pour que le Cogensys Judgment Software présenté précédemment soit fiable. Par contre, selon le document *Concept Learning and Heuristic Classification in Weak Theory Domains* [6], CL-Protos atteindrait ce même niveau de fiabilité après seulement 200 exemples résolus. Il nous a été impossible de faire des tests pour vérifier ces assertions, et nous nous en remettons à la bonne foi des parties impliquées.

L'entrée des nouveaux cas se fait interactivement, mais CL-Protos n'utilise pas d'illustrations graphiques pour afficher ses connaissances acquises. Son interrogation est originale, se faisant en pseudo-anglais, et est très souple car elle permet de définir des synonymes, des approximations conditionnelles, et autres. Le logiciel produit en sortie la liste des cas avec leurs importances prototypiques et leurs attributs tels que le logiciel les a détectés. Cependant, comme l'indique la figure 21 (sortie par CL-Protos d'un réseau de cas tel que celui présenté à la figure 6), une telle liste n'est pas très explicite.

Étant non commercialisé et produit par une équipe de recherche, CL-Protos n'est pas équipé d'une interface sophistiquée pour l'utilisateur. Tout se déroule sur une seule fenêtre en mode texte avec menu. Les concepteurs l'ont voulu ainsi pour conserver le logiciel portable, car ils désirent le rendre accessible à quiconque veut l'étudier. Malheureusement, ceci a des conséquences sur la facilité de compréhension et d'apprentissage du logiciel. Alors qu'avec NEXTRA il ne faut pas plus de 10 minutes, il faut compter au moins trois heures avec CL-Protos pour visualiser tous les sous-menus hiérarchisés. Nous sommes convaincus qu'avec une interface conviviale similaire à celle de NEXTRA, toutes ces difficultés s'estomperaient. La documentation accompagnant le logiciel est conçue pour un lecteur s'y connaissant déjà en informatique et en logiciels d'acquisition de connaissances; elle est claire et structurée.

2.2 Synthèse comparative des logiciels évalués

Nous présentons dans cette section une comparaison sous forme de tableau récapitulatif et une synthèse de notre évaluation des sept LAC exposés précédemment. Les critères utilisés pour l'évaluation et dans le tableau récapitulatif sont d'abord exposés.

2.2.1 Critères utilisés pour l'évaluation

Voici un survol des critères utilisés pour notre évaluation. Ils sont regroupés en quatre catégories : critères de clientèle, structure du logiciel, nature des entrées et sorties et la convivialité du LAC.

- Critères de clientèle

Ces critères s'adressent à un acquéreur potentiel et lui permettent de savoir si le logiciel lui convient :

. Commercialisé : indique si le logiciel est disponible sur le marché. Si c'est le cas, nous indiquons son coût approximatif (taxe exclue) de même que les services de support inclus.

. Besoin d'un cogniticien : indique s'il est nécessaire de recourir à un cogniticien pour manipuler le logiciel. Dans le cas d'une réponse négative, l'expert peut directement utiliser le logiciel.

. Environnement : indique les besoins matériels (type d'ordinateur, système d'exploitation spécifique, équipement particulier, et autres) et logiciels (langages nécessaires) pour que le LAC fonctionne.

. Applications antérieures connues : les domaines dans lesquels le logiciel a servi à développer avec succès une base de connaissance. Toutefois, ceci ne limite aucunement le logiciel à certains domaines d'expertise.

- Structure du logiciel

Ces critères résument le type d'architecture, le type de théorie, le format de la BC générée. La section 2.1 décrit de façon détaillée chaque logiciel.

. Type d'architecture : séparée ou intégrée. Cette donnée est importante car si l'architecture est séparée, il faut se procurer un GSE compatible avec la base de connaissances générée.

. Format de la base de connaissances générée : règles de production ou cas. Si la base générée est sous forme de règles, nous indiquons si les règles sont structurées en arbre de décision ou en réseau de décision. Si la base générée est sous forme de cas, nous indiquons si les cas sont organisés de quelque façon. Cette donnée peut avoir une influence sur les performances dans le cas d'une vaste base de connaissances.

. Type d'applications : classification ou diagnostic (la classification étant un sous-ensemble du diagnostic). La classification possède typiquement un ensemble fixe d'attributs qui permettent de distinguer les objets de la classe, alors que le diagnostic comporte des attributs qui varient d'une conclusion à l'autre. Il est toujours possible de traiter un problème de diagnostic comme un problème de classification, mais l'arbre de décision obtenu peut devenir immense.

. Type de théorie : une expertise à théorie forte utilise des règles connues de l'expert et permettant aisément de prendre une décision fiable. Par opposition, une expertise à théorie faible est celle où l'intuition et l'expérience sont grandement utilisées.

- Nature des entrées et sorties

Cette section décrit les modalités d'entrées, de manipulations et de sorties des informations traitées par le logiciel.

. Entrées : les entrées peuvent se faire par l'utilisateur en mode interactif ou le logiciel peut aller chercher les données via des processus automatiques (manipulation de bases de données, traitement en lot, etc.).

- Manipulation : indique les techniques d'interrogation utilisées, l'utilisation du graphisme par le logiciel pour présenter ses connaissances et l'efficacité du mode d'interrogation.

- Sorties : qualité de la visualisation de la base de connaissances, optimisation des règles (la performance du résultat en dépend) et ouverture sur l'extérieur (accès à des logiciels autres que l'exportation vers des GSE).

- Convivialité

Dans cette section, nous décrivons d'abord la facilité à apprendre le logiciel, à l'utiliser et à mettre à jour. Ensuite, sont évaluées les qualités de l'interface et de la documentation. Enfin, nous commentons la gestion des messages d'erreurs et les "bugs" observés au niveau du logiciel.

2.2.2 Tableau comparatif des sept logiciels

1. Critères de clientèle	Sophos	KAT	CAMEO	IXL	NEXTRA	Cogensys	CL-Protos
Commercialisé	oui	oui	non	oui	oui	oui	non
si oui : coût	1500 $	400 $ US	-	500 $ US	4 000 $	250 000 $ US	-
service inclus	90 jrs (téléphone)	aux clients de ADL	-	non	90 jours	inclus dans le prix	-
Besoin d'un cogniticien	non	non	oui	non	non	non	non
Environnement matériel	IBM - DOS	Macintosh	Macintosh	IBM XT+	AT+, Macintosh II+, VAX, Sun	IBM	Sun3
Environnement logiciel	nil	nil	HyperCard	fichiers dBase, Lotus ou ascii	nil	nil	Kyoto Common Lisp
Applications antérieures connues	prêts bancaires	nil	autorisation d'assurances	ligne de production	verres de contact	assurances	nil

2. Structure de l'outil	Sophos	Kat	Cameo	IXL	Nextra	Cogensys	CL-Protos
Architecture (type)	séparée	séparée	séparée	séparée	séparée	intégrée	intégrée
Si séparée, compatible avec GSE	VP-Expert, Intelligence Service 2 Level 5, texte	Level 5	KEE, NEXPERT *Object*, Macbrain, Neuralworks	texte, respectant la norme de GSE (voir 2.1.4)	Nexpert *Object*	-	-
Base de connaissances générée à partir de	règles	règles	règles/objets/processus	règles	règles	cas	cas
Si règles, leur structure	arbre	arbre(s)	-	en vrac	réseau	-	-
Si cas, ordonnancement	-	-	-	-	-	aucun	pointeurs
Type d'applications	classification	diagnostic	diagnostic	diagnostic	diagnostic	diagnostic	diagnostic
Type d'expertise	forte	forte	forte	faible	faible ou forte	faible	faible

3. Entrées - Sorties	Sophos	KAT	CAMEO	IXL	NEXTRA	Cogensys	CL-Protos
Entrées							
mode	interactif	interactif	interactif	automatique	interactif	?	interactif
type de données	théorie	théorie	théorie	bases de données	exemples décrits par des attributs	exemples résolus	exemples résolus
Manipulation							
technique d'interrogation	réponses à toutes les combinaisons d'attributs	remplissage de pages graphiques	aucune	choix de réponses	grilles d'attributs	choix de réponse	questions du logiciel
interrogations variées	non	non	-	non	oui	non	non
graphisme	oui, limité	oui	oui	non	oui	non	non
efficacité	non	non	oui	oui	oui	oui	oui
Sorties							
bonne visualisation	non	oui	oui	non	oui	non	non
optimisation des règles	non	feuilles seulement	non	non	oui	oui	oui
ouverture sur l'extérieur	non	non	texte (transcription d'entrevues)	bases de données	non	oui, accès à réseau	non

4. Convivialité	Sophos	Kat	Cameo	IXL	Nextra	Cogensys	CL-Protos
facile à apprendre	oui	oui	oui	non	oui	oui	moyen
facile à utiliser	non	oui	oui	oui	oui	oui	oui
facile à mettre à jour	non	oui	oui	n.a.	oui	oui	oui
qualité de l'interface	faible	bonne	excellente	moyenne	excellente	moyenne	faible
gestion des messages et erreurs	moyenne	bonne	bonne	?	très bonne	?	moyenne
qualité de la documentation	moyenne	bonne	bonne	?	très bonne	?	très bonne
'bug' observé	aucun détecté	aucun détecté	aucun détecté	?	aucun détecté	?	aucun détecté

2.2.3 Conclusion de notre évaluation

En conclusion, nous mentionnons d'abord les avantages et désavantages principaux que nous avons observés pour chaun des logiciels :

Sophos :

- d'apprentissage rapide;
- adaptable à plusieurs GSE;
- utile pour des problèmes simples de classification;
- l'expert peut travailler directement avec le logiciel;

mais,

- désagréable à utiliser (interface faible et interrogations pénibles);
- aucune optimisation de la base de connaissances;
- expertise à théorie forte seulement;
- classification surtout, mais diagnostic possible.

KAT :

- d'apprentissage rapide;
- puissance du graphisme, très visuel;
- exploite au maximum les possibilités de Level5;
- traite plusieurs buts et plusieurs arbres;

mais,

- expertise à théorie forte seulement;
- grand nombre de facteurs à remplir.

CAMEO :

- facilité d'apprentissage et d'utilisation;
- graphisme sophistiqué;
- développement rapide;

- générique (exportable vers de nombreuses coquilles);
- traite les données (modèle de données) et les opérations (modèle de processus);
- permet l'analyse de transcriptions;

mais,

- expertise à théorie forte seulement;
- lourdeur de la représentation (taille du modèle résultant).

IXL :

- infère les règles sur des bases de données existantes;
- permet de développer des politiques;
- expertise à théorie faible pouvant se traduire en une théorie forte;

mais,

- n'exporte pas ses règles vers des systèmes d'aide à la décision;
- analyse uniquement statistique de bases de données existantes.

NEXTRA :

- très visuel;
- infère lui-même les règles sur des grilles d'attributs;
- permet de développer des politiques;
- expertise à théorie faible ou forte;
- environnement multiexpert;

mais,

- les exemples ne peuvent provenir d'une base de données.

Cogensys Judgment Software :

- facile d'utilisation;
- large environnement de travail idéal pour l'entreprise;
- expertise à théorie forte ou faible;

mais,

- aucune visualisation possible;
- inaccessibilité du mode de fonctionnement du logiciel.

CL-Protos:

- apprentissage rapide sur un nombre limité de cas;
- expertise à théorie faible;
- basé sur des recherches récentes et prometteuses sur l'apprentissage par exemples;

mais,

- interface peu développée.

Pour choisir un LAC qui soit profitable, il est important d'examiner le type d'expertise. Si l'expertise à élucider est à théorie faible et qu'une théorie forte est désirée, nous recommandons NEXTRA. Malheureusement, ce dernier n'accepte pas les exemples résolus par l'intermédiaire des bases de données. S'il existe de vastes bases de données sur le domaine à élucider, il serait préférable d'utiliser IXL.

Si l'expertise est à théorie faible et qu'aucune théorie forte n'est vraiment désirée, il est préférable d'avoir recours à un logiciel par cas. Les candidats possibles sont Cogensys Judgment Software et CL-Protos. Il faut effectuer le choix selon les avantages et les désavantages respectifs de chacun.

Si l'expertise est à théorie forte et que l'expert la maîtrise bien, alors un simple logiciel de formalisation des connaissances suffit, comme par exemple KAT ou Sophos. CAMEO, plus générique, peut être d'une grande utilité et offre des caractéristiques plus variées (objets, structure des règles et contrôle).

CONCLUSION

Il est sans doute impossible de découvrir ou d'expliciter toutes les connaissances d'un vaste domaine d'expertise à moins que ces connaissances ne soient acquises automatiquement. Le but des recherches sur les LAC consiste donc à définir des connaissances pour découvrir et utiliser les connaissances du domaine : les méta-connaissances. La difficulté principale se situe alors au niveau du démarrage : il faut graduellement atteindre une "masse critique" qui permette au logiciel de continuer seul.

Une difficulté majeure à surmonter au niveau des LAC est la prise en considération de l'aspect sémantique de la connaissance, un élément primordial bien souvent mis de côté. En effet, les LAC courants ne sont actuellement que des outils (souvent graphiques) contraignant l'expert à formaliser son savoir sous forme d'arbres ou de réseaux d'objets et d'attributs.

La panoplie des logiciels étudiés dans ce rapport n'est pas complète, mais est cependant représentative des disponibilités actuelles. Actuellement, le cogniticien joue encore un rôle important, même si ce n'est qu'un rôle de supervision et de mise en œuvre du LAC. Toutefois, nous pouvons nous demander quand verra le jour un système (expert) pouvant réaliser de manière autonome et complète l'exploration, l'analyse et la synthèse du champ de connaissances ciblé, tout en se comportant de façon naturelle avec l'expert. Un tel logiciel n'est certes pas encore réalisable car il doit réunir de nombreuses qualités réservées (au moins pour l'instant) à l'humain, comme le sens commun, le sens des valeurs, les connaissances sémantiques, etc [11].

Nous présentons pour terminer un ensemble d'avenues potentielles de recherche :

- développement d'un LAC qui, d'une part, peut parcourir seul le domaine de ses connaissances et, d'autre part, peut produire des scénarios de type *what if*, pour trouver de nouvelles règles, faits, ou méta-connaissances (c'est le logiciel qui initie l'interrogation et non plus le contraire);

- explorer les possibilités de l'apprentissage de concepts par généralisation ou spécialisation, par analogie, par induction, et autres;

- utilisation de définitions et de cas négatifs pour raffiner les connaissances, permettant par exemple à l'expert des énoncés du type "Je ne sais pas comment le décrire. Ce n'est certainement pas comme ça, mais ça ressemble à ceci";

- la représentation des connaissances est un domaine lié à l'avancement des LAC. Elle peut notamment explorer la définition de formats de connaissances ou de techniques permettant les traitements nécessaires dans les trois phases de développement de la BC (voir section 1.2);

- le domaine des réseaux neuronaux (modèles connexionistes) est encore relativement nouveau; les propriétés remarquables de tels systèmes concernent justement l'apprentissage et la généralisation;

- le traitement de la langue naturelle (et synthèse/reconnaissance de parole) est aussi un domaine connexe dont bénéficierait les LAC;

- le domaine des bases de données, en particulier les nouvelles approches dites intelligentes, orientées objet, déductives et hypertexte, est aussi relié aux LAC; en effet, cette avenue permet de profiter de connaissances existantes enregistrées implicitement dans les bases de données et vise aussi à organiser les connaissances pour en avoir un accès efficace (requêtes intelligentes);

- le développement de méthodes et de logiciels pour la coordination du travail en groupes (2 à 5 personnes, ou même plus), probablement par le biais d'un réseau de télécommunications;

- l'exploration du domaine psychologique (cognitif, en particulier) pour le développement de méthodes d'interrogation et d'enregistrement des connaissances.

GLOSSAIRE

BC : base de connaissances, soit le résultat de la formalisation des connaissances de l'expert utilisable par un GSE;

cas : exemple typique d'une situation avec sa solution;

GSE : générateur de systèmes experts ou outil de développement de systèmes experts;

LAC : logiciel d'acquisition des connaissances;

SE : système expert ou système à base de connaissances.

RÉFÉRENCES

[1] Sandra Marcus (ed.), *Automating Knowledge Acquisition for Expert Systems*, Boston : Kluwer Academic Publishers, c1988, xvi, 270 p. (The Kluwer international series in engineering and computer science, SECS57.)

[2] Ray Bareiss, *Exemplar-based knowledge acquisition : a unified approach to concept representation, classification, and learning*, Boston : Academic Press, c1989, xii, 169 p. (Perspectives in artificial intelligence, 2.)

[3] E. R. Bareiss, B. W. Porter et C. C. Wier, "Protos : An Exemplar-Based Learning Apprentice", In John R. Anderson, *Machine Learning : An Artificial Intelligence Approach*, vol. 3, Los Altos, Calif. : Morgan Kaufmann Publishers Inc., 1988.

[4] *CAMEO*. ADL (Arthur D. Little), Acorn Park, Cambridge, Massachussett, USA 02140. Mr William Jones, (617) 864 5770 (ext. 2482), (617) 776 2472.

[5] J. Cendrowska, "PRISM : An algorithm for inducing modular rules", *International Journal of Man-Machine Studies*, vol. 27, no 4 (1987), pp. 349-70.

[6] *CL-PROTOS*. Vanderbilt University, Box 155, Station B, Nashville, Tennessee, USA 37235. M. Ray Bareiss, (615) 322 3233, bareiss@vuse.vanderbilt.edu.

[7] *COGENSYS Judgment Software*. Cogensys, 11 077 North Torrey Pines Road, La Jolla, California, USA 92037. Mr. Nick Van Der Shalley, (619) 458 1500.

[8] Hubert L. Dreyfus, *What computers can't do*. Harper Colophon, 1979.

[9] Brian R. Gaines et Mildred L. G. Shaw, "A Methodology for Recognizing Consensus, Correspondence, Conflict, and Contrast in a Knowledge Acquisition System". In *Proceedings of the 3rd Knowledge Acquisition for Knowledge-Based Systems Workshop : Banff Conference Centre, Banff, Alberta, Canada, November 6-11, 1988*.

[10] Anna Hart, *Knowledge Acquisition for Expert Systems*. New York : Mc Graw Hill, 1986, 180 p.

[11] J.-L. Lemoigne et al., *Intelligence des mécanismes, mécanismes de l'intelligence*, Éd. Fayard, 1986.

[12] *IXL*. Intelligence Ware Inc., 9800 S. Sepulveda Blvd., Suite 730, Los Angeles, California, USA 90045. Mr Chuck Babbitt, (213) 417 8896.

[13] *KAT*. Computer Aided Knowledge Engineering Tools, 26 Sala Court, Spring Valley, New York, USA 10977. Mr Gary S. Ribar, (201) 307 7866.

[14] G. A. Kelly, *The Psychology of Personal Constructs*, New York : Norton, 1955.

[15] "Knowledge Acquisition", *Expert Systems Strategies*, vol. 5, no. 7 (1989), p. 1-6.

[16] "Knowledge Acquisition Tools", *Expert Systems Strategies*, vol. 5, no. 7 (1989), p. 7-15.

[17] John H. Boose, Brian R. Gaines (ed.), *Knowledge Acquisition for Knowledge Based Systems*, Toronto : Academic Press, 1988, xix, 355 p. (Knowledge based systems, 1.)

[18] John H. Boose, Brian R. Gaines (ed.), *Knowledge Acquisition Tools for Knowledge Based Systems*, Toronto : Academic Press, 1988. (Knowledge Based Systems, 2.)

[19] George F. Luger et William A. Stubblefield. *Artificial intelligence and the design of expert systems*, Redwood City, Calif. : Benjamin/Cummings Pub. Co., c1989, xxv, 660 p., 1989. (Benjamin/Cummings series in artificial intelligence.)

[20] Karen L. McGraw et Karan Harbison-Briggs, *Knowledge acquisition : Principles and guidelines* , Englewood Cliffs, N.J. : Prentice Hall, c1989, xvi, 376 p.

[21] *NEXTRA*. Neuron Data, 444 High Street, Palo Alto, California, USA 94301. Mr Rico Bumbaca, (415) 321 4488.

[22] K. Parsaye et M. Chignell, *Expert Systems for Experts,* Toronto : John Wiley & Sons, 1988, p. 327-374.

[23] J.R. Quinlan "Induction of decision trees", *Machine Learning*, 1986.

[24] Mildred L. G. Shaw, *On becoming a personal scientist : Interactive computer elicitation of personal models of the world*, New York : Academic Press, 1980, xvi, 322 p. (Computers and people series, 1.)

[25] P. Slater, *Explorations of Intrapersonal Space : volume 1*. London : John Wiley, 1976.

[26] *SOPHOS*. Cognisys, 1755 est, boul. René-Lévesque, Montréal, Québec, Canada, H2K 4P6. Mme Fabienne Fayad, (514) 599 5714.